Méthodes de vérification de bases de connaissances

Philippe Lafon

ECOLE NATIONALE DES PONTS ET CHAUSSEES

THESE

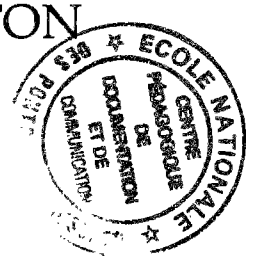
présentée pour obtenir

le titre de DOCTEUR de l'E.N.P.C.

Spécialité: MATHEMATIQUES et INFORMATIQUE

par

Philippe LAFON



SUJET : Méthodes de Vérification de Bases de Connaissances

soutenue le 18 Décembre 1991 devant le jury composé de :

M. Michel GONDRAN	Président
Mme Luisa ITURRIOZ	Rapporteur
M. Marc AYEL	Rapporteur
M. René LALEMENT	
M. Jean-Luc DORMOY	

REMERCIEMENTS

Monsieur GONDRAN m'a accueilli au sein de la Direction des Etudes et Recherches de l'EDF. Je le remercie d'avoir su réunir les conditions qui m'ont permis de faire cette thèse, et de me faire l'honneur d'être Président du jury.

Je remercie Madame ITURRIOZ de s'être intéressée à mes travaux. Sa contribution à ce travail m'a permis d'effectuer des corrections utiles à la qualité de ce manuscrit. De plus, sa gentillesse et son ouverture d'esprit, à un moment où son emploi du temps était particulièrement chargé, ont révélé une femme de cœur. Je suis donc doublement fier qu'elle participe au jury de cette thèse.

Je remercie aussi Monsieur LALEMENT d'avoir accepté de faire partie du jury. Je découvris grâce à ces cours, il y a quatre ans, les voix charmeuses et redoutables de la Logique. Ce travail lui doit donc beaucoup. Qu'il soit présent à son aboutissement m'honore, et, symboliquement, me parait ... Logique.

Jean-Luc DORMOY fût, pour moi comme pour beaucoup d'autres, un guide éclairé et un animateur de ma recherche. Il me donna des idées, de la confiance, et un peu de la passion qui l'anime. Sa présence parmi les membres du jury m'honore grandement.

C'est un honneur rare, je crois, de pouvoir compter, dans son jury de thèse, le spécialiste français du domaine de recherche que l'on a choisi. La présence de Marc AYEL dans le jury m'offre cette chance, qui est aussi un précieux avantage. Ses remarques, en tant que rapporteur, l'ont pleinement prouvé. Je l'en remercie vivement.

J'ai peu le goût des effusions publiques ("Asinus asinum fricat ..."). Aussi ne trouvera-t-on pas ici une liste détaillée de tous les collègues, amis, parents, et autres bestioles, "ayant apporté une aide précieuse dans la phase finale ou durant cette thèse". Tout le monde sait que la recherche est un travail collectif, et plutôt pénible pour l'entourage du chercheur. Je ne peux toutefois m'empêcher d'exprimer ma profonde gratitude envers Jean-Luc, Jean-Philippe et François. Je sais que cette thèse est peu de chose. Sans eux, elle ne serait rien.

PLAN

INTRODUCTION

Il est connu que l'approche scientifique d'un problème se ramène principalement à deux phases: la **modélisation** et la **résolution**. Ces deux phases sont évidemment liées, puisqu'on ne modélise pas un problème sous une forme pour laquelle on n'a pas de méthode de résolution.

Jusqu'à présent, la modélisation d'un problème était effectuée dans le langage mathématique ou sous forme d'une représentation graphique. La nouveauté qu'a apportée, au niveau modélisation, l'Intelligence Artificielle a été dans les années 50 d'ajouter la représentation symbolique aux deux représentations précédentes et surtout, pour ce qui va nous intéresser ici, de rendre **modulaire** cette représentation symbolique en créant le concept de **base de connaissances**. Il est alors devenu possible de spécifier progressivement son problème en construisant sa base de connaissances.

Au niveau de la résolution, la nouveauté qu'a apportée l'Intelligence Artificielle fût, avec le concept de **système expert**, de **séparer complètement la méthode de résolution de la base de connaissances**. Dans ce schéma, les connaissances d'un domaine sont considérées comme des données pour le programme informatique, qui n'est plus qu'une simulation d'un raisonnement sur cette connaissance. Ce programme - le moteur d'inférence - est une entité universelle, indépendante du problème considéré.

Toutefois, aujourd'hui, cette idée est plus riche de potentialités que de réalisations effectives. En effet, si l'intérêt soulevé par cette nouvelle approche pour la résolution de problèmes complexes fût et demeure considérable, le nombre de systèmes experts commercialisés est, en comparaison, dérisoire.

La raison de ce décalage tient essentiellement à l'**absence de méthodes confirmées de développement, de vérification et de validation des systèmes experts**. De cette lacune résulte un problème rédhibitoire d'évaluation de la qualité et de la fiabilité du produit informatique réalisé, avec parfois des conséquences dramatiques sur l'achèvement du projet: "s'il est très facile de réaliser une maquette en quelques mois (voire quelque jours), il est parfois impossible de réaliser un système fiable et robuste en plusieurs années"(extrait du rapport du conseil scientifique d'EDF au sujet des études d'Intelligence Artificielle à EDF [EDF88]).

Ces problèmes cruciaux: développement, vérification, validation de programmes, se sont bien sûr également posés en informatique classique. Ils ont donné naissance à un domaine de recherche à part entière, le génie logiciel. Son équivalent en intelligence artificielle, **le génie de la connaissance** en est à ses balbutiements. **Le travail présenté dans cette thèse se situe dans le cadre général du génie de la connaissance**. Plus précisément, quand, dans un système expert, le formalisme utilisé pour représenter les connaissances est celui des règles de

production, on parle de système à base de règles. **Notre travail concerne la vérification de systèmes experts à base de règles.**

La vérification d'un système expert comporte naturellement plusieurs facettes. Nous nous sommes intéressé à la plus élémentaire d'entre elles, qui correspond à la **vérification que la connaissance contenue dans la base de connaissance n'est pas contradictoire.** Cette vérification est connue sous le nom de **vérification de la cohérence d'une base de connaissance.**

Différentes logiques sont utilisées comme fondement du formalisme des règles des systèmes experts à base de règles. Quand les règles ne font intervenir que des entités constantes, le système est dit d'ordre zéro, par référence avec la logique d'ordre zéro. En effet on peut alors faire une analogie entre entités de la base de règles et variables propositionnelles, et considérer la base de règles comme un ensemble d'axiomes d'un système formel ayant la mêmes règle de déduction que le système formel de la logique des propositions (le Modus Ponens). Le problème de la non-contradiction de la base de connaissances peut être vu comme un problème de consistance de théories.
On est donc ici dans le cas où le formalisme de la base de connaissances est suffisamment proche d'un modèle mathématique bien connu pour pouvoir résoudre la question de la présence de contradiction dans les règles de manière entièrement automatique, c'est-à-dire sans aucune intervention du concepteur. **On est en effet capable de donner à la machine un algorithme qui lui permette de trouver seule les démonstrations contradictoires possibles à partir de la base de connaissances** (c'est une conséquence de ce que JL LAURIERE appelle à juste titre la lisibilité, par la machine, des bases de règles, cf [LAURIER87]).

Dans le cas des systèmes experts d'ordre zéro plus, c'est-à-dire quand la syntaxe des règles est basée sur des expressions du type (entité comparateur valeur), le rapprochement avec la logique des propositions est aussi possible, et le processus de contrôle de l'absence de contradiction se ramène assez aisément à celui d'un système d'ordre zéro.

Toutefois, l'évolution des systèmes experts vers la résolution de problèmes plus difficiles a conduit à une complexification des modèles de représentation des connaissances. Ces nouveaux formalismes, riches et souples, permettent d'accueillir des raisonnements de natures diverses. Mais la complexité des bases de connaissances ainsi réalisées pose de manière nouvelle le problème de la contradiction dans les bases de connaissances. Ici, il n'est plus question de rapprochement évident avec la logique - même si on utilise volontiers sa terminologie en parlant de variables, de prédicats, de système d'ordre Un, ... - ni de contradictions "explicites" dans les règles. **Le concept de contradiction doit ici se définir comme le non-respect par la base de règles d'une des propriétés émergeant du modèle conceptuel de la base de règles,** ce modèle pouvant être vu comme une spécification partielle de la modélisation réalisée dans la base de connaissances.

Le travail que nous allons présenter s'intéresse à la vérification de l'absence de contradiction dans des systèmes experts à base de règles d'ordre Zéro Plus ou d'ordre Un. Ce même problème recouvre deux réalités très différentes suivant l'ordre du système qu'il s'agit de contrôler. Nous avons donc réalisé deux outils de contrôle. Chacun sera décrit dans une partie séparée de ce document.

Plan du document:
Plus précisément, nous étudierons successivement :

- dans le chapitre 1, les différentes approches existantes sur le problème de la vérification de la non contradiction dans les systèmes à base de règles, puis nous reviendrons sur notre démarche et préciserons nos objectifs,

- dans la partie 1: le problème de la cohérence de bases de règles d'ordre Zéro Plus. Plus précisément, nous étudierons:
  - dans le chapitre 2, le passage du système formel directement inspiré de la base de règles à un système formel plus simple, et définirons dans ce cadre la propriété de cohérence, tant sur le plan syntaxique que sémantique;
  - dans le chapitre 3, nous dévoilerons l'algorithme de vérification de la cohérence implémenté, et établirons la preuve, tant syntaxiquement que sémantiquement, de sa justesse;
  - dans le chapitre 4, nous exposerons l'aspect pratique des théories développées dans les deux chapitres précédents, en présentant l'outil informatique réalisé: MELOMIDIA, ses résultats, et nous le comparerons à son "grand frère": MELODIA.

- dans la partie 2: le problème de la cohérence de bases de règles d'ordre Un
  Plus précisément, nous exposerons:
  - dans le chapitre 5, la syntaxe des règles concernées, puis nous continuerons par un survol du système réalisé: VAUBAN1, et enfin nous montrerons un premier type de contrôle effectué par le système;
  - dans le chapitre 6, le modèle conceptuel de base de connaissances que nous avons défini, puis les vérifications de la base de connaissances possibles en regard de ce modèle.

Enfin, nous conclurons par un résumé des travaux effectués et par une réflexion sur leurs prolongements éventuels.

CHAPITRE I: Vérification de Bases de Connaissances Présentation

1) Vérification de bases de connaissances : Etude de l'existant

a) Vérification, Validation et Test

La recherche dans le domaine du contrôle des systèmes à base de connaissances est en plein essor. Ce développement est significatif d'une maturité certaine sur quelques aspects de ce problème. En effet, une quantité assez importante de travaux sur ce sujet est maintenant connue. Toutefois, il y a souvent confusion entre les concepts de Vérification, de Validation et de Test de systèmes à base de connaissances. Nous allons donc, avant de décrire les différentes approches réalisées jusqu'à présent dans le domaine de la vérification de systèmes à base de connaissances, préciser quel est l'objectif de cette opération et la situer par rapport à celles de Validation et de Test[1].

L'opération de vérification est, d'après la norme IEEE (n° 729-1983): "L'opération de contrôle que le résultat d'une phase donnée du développement d'un logiciel respecte les spécifications établies pendant la phase précédente". Autrement dit, cette opération consiste à s'assurer que l'on a pas perdu ou rajouté d'informations d'une étape à l'autre du développement d'un logiciel.

Suivant les mêmes sources, l'opération de validation se définit de la façon suivante: "L'opération d'évaluation d'un logiciel à la fin de la phase de développement garantissant sa conformité aux spécifications initiales".

Quant au Test, il désigne une technique couramment utilisé pour vérifier ou valider un système. La principale question liée à l'utilisation de cette technique étant: "Quand peut-on dire que le système est suffisamment testé?" C'est-à-dire sous quelles conditions peut-on conclure, à partir du résultats de tests, que le système est vérifié ou valides.

En résumé, les deux concepts importants sont celui de Vérification et celui de Validation. Le premier correspond au fait de poser la question: "Est-ce que (le fonctionnement de) mon système est bon ?". Le deuxième correspond, lui, à la question: "Est-ce que l'on a construit le bon système ?"[2].
Le travail de vérification est mené par le concepteur du système, celui de validation par le destinataire du système.

Les travaux que nous allons maintenant présenter (et ceux que nous avons réalisés, que nous présenterons dans les chapitres suivants) se situent clairement dans la première catégorie.

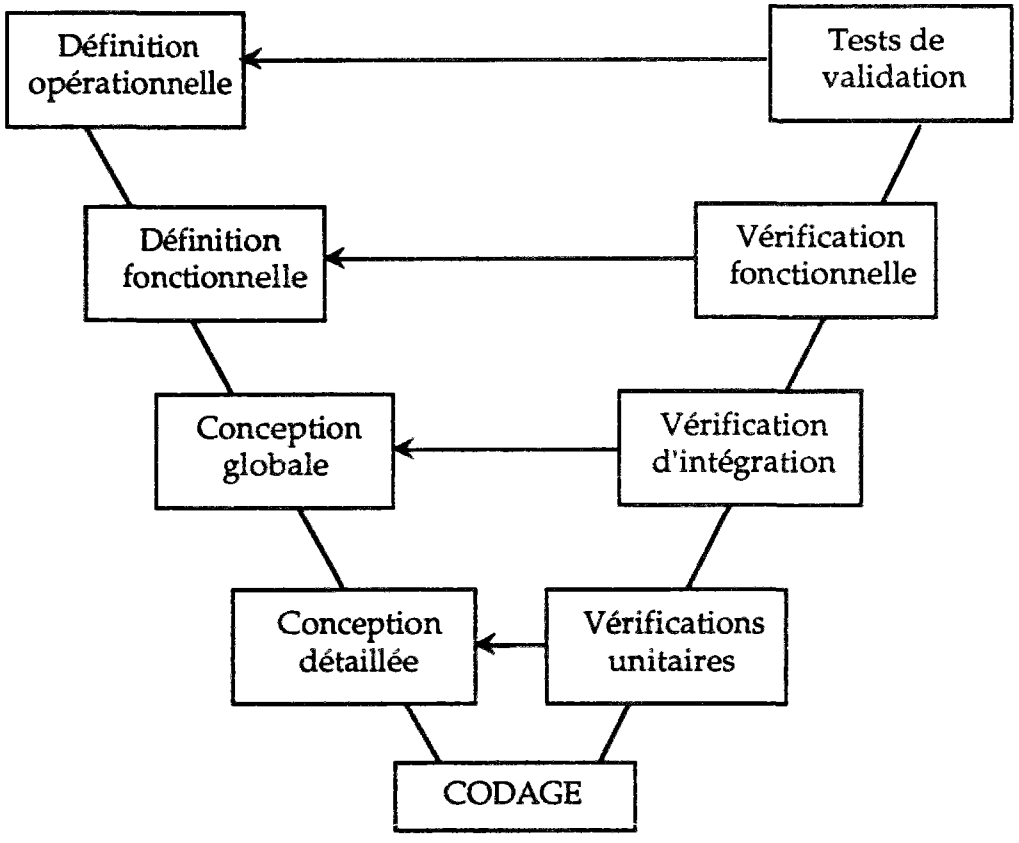
---

[1] Il n'y a toutefois pas consensus sur le sujet. Le point de vue que nous adoptons ici est celui développé dans [AYEL&L91a] et [VIGNOL91]. On peut le confronter aux classifications exposées dans [BRU&Al91] et [HOPP&Al91].

[2] Ce que les anglo-saxons expriment malicieusement par:
"Is the system right?" et "Is the right system?" [O'KEE&Al87]

b) L'approche "Génie Logiciel"

La construction de logiciels en programmation "classique" repose sur des méthodes dans lesquelles le processus de vérification est intégré. Ces méthodes consistent à concevoir le logiciel suivant une démarche "en V" (d'après [MUENIE89]:



La première branche du V représente la démarche de construction des spécifications du problème. On part d'une définition très générale et très "abstraite" du problème pour arriver progressivement, conceptuellement, à la conception détaillée du produit informatique.

Commence alors la "remontée", sur la branche "programmation" du V du particulier (écriture des modules) vers le général (écriture du "monitoring"). A chaque étape de la programmation, on teste si le code respecte la spécification correspondante.

Partant de ce schéma,certains auteurs proposent des méthodes de vérification pour les Systèmes à Base de Connaissances (SBC).

Ainsi C. GREEN et M. KEYES, dans [GREE&K87], insistent sur l'absolue nécessité d'avoir des spécifications du SBC sur au moins deux niveaux , et donnent quelques pistes pour les écrire. Ils mentionnent ensuite la liste des vérifications nécessaires à une bonne pénétration de la technologie système expert dans l'industrie, rajoutant aux vérifications "classiques" celles de

- l'analyse de l'interaction entre îlots de connaissances
- l'analyse de la pertinence des règles, en terme de connaissances

- l'analyse de l'incertitude, dans le cas où le système manipule des coefficients de vraisemblance des faits.

Remarquons que pour ces auteurs, il est exclu que ces vérifications soient automatisées. Elles doivent être effectuées par une équipe de spécialistes ("V&V practionners") extérieure à celle de conception, et bénéficiant d'une autorité sur celle-ci qui lui permette d'exiger, par exemple, la rédaction des spécifications.

Dans le même ordre d'idées, J.R. GEISMAN et R.D. SCHULTZ, dans [GEISS&S90], défendent la nécessité du développement d'un SBC en 6 étapes, la première, spécifique à l'IA, étant la réalisation d'un prototype informatique à partir duquel seront écrites les spécifications, et préparés les tests, du système futur. Là encore, l'accent est mis sur l'importance d'un développement à partir de spécifications. Constatant seulement la difficulté de débuter le projet par l'écriture des spécification, les auteurs proposent de commencer par un codage "expérimental" du système futur.

En conclusion, ces approches semblent adaptés à de très gros projets, ou à des projets très "sensibles". Les auteurs ne cherchent pas à faire preuve d'originalité mais au contraire à se réfugier derrière des méthodes et des techniques qui ont été reconnus valides. Leur principal souci est l'acceptation du SBC par le client ou les autorités de tutelle. Le profil est donc volontairement bas, l'originalité étant, on le sait, en général mal perçue. L'impression générale peut donc se résumer de la façon suivante : "On sait que ça ne convient pas très bien, mais les autorités ont déjà accordé leur confiance à ces méthodes - sous-entendu "dans des cas où elles ne s'appliquaient pas davantage" - donc ...".

c) L'approche "Méthode de Conception de Bases de Connaissances"

Partant du constat établi ci-dessus, à savoir: "les méthodes traditionnelles de développement de logiciels s'appliquent mal au développement des logiciels d'IA", des équipes ont repris à zéro le problème. Elles ont conçu des méthodologies de développement de SBC à la fois concurrentes et complémentaires. Nous allons nous intéresser aux deux méthodologies les plus connues, en les étudiant sur le plan de la vérification.

La méthode K.O.D. ([VOGEL88]) s'intéresse à la première phase de l'élaboration d'un SBC. Celle-ci, dite de Spécification, précède celle de Conception, qui précède elle-même celle de Réalisation. Elle consiste, en partant des interviews de l'expert, à construire une formulation structurée du problème, et des connaissances sur le problème (cette formulation étant indépendante du choix du générateur de SBC). Cette tache prend la forme de l'écriture de quatre modèles successifs: le modèle textuel, le modèle pratique, le modèle cognitif et enfin le modèle informatique. Le travail de Vérification concerne donc les spécifications produites à ces quatre niveaux. Il est assurée grâce à l'utilisation d'un vocabulaire précis auquel est adjoint des mécanismes de contrôle syntaxique, sémantique et structural.

Plus précisément, la méthode K.O.D. a donné lieu à la réalisation d'une machine, la K-Station, qui aide le concepteur à l'utiliser. Le travail de vérification effectué par la K-Station consiste en :

- la vérification automatique les liens entre niveaux conceptuels (gestion automatique des références croisées entre entités K.O.D.)
- la construction et l'actualisation automatiquement d'un dictionnaire de données
- l'analyse et la qualification des objets K.O.D. (ce dernier point n'étant pas développé dans la documentation à notre disposition, nous le donnons sous toute réserve)

En conclusion, l'approche de la "vérification" (au sens "faire de bons systèmes") de K.O.D. semble se résumer à:

- donner à l'utilisateur des outils puissants de visualisation, style hypertexte, de son travail, de façon à ce qu'il lui soit plus facile de vérifier lui-même la cohérence de son modèle,
- définir un cadre rigoureusement structuré d'écriture de SBC, de façon d'une part à ce que l'utilisateur réfléchisse mieux - donc se trompe moins , d'autre part à effectuer quelques contrôles liés à l'utilisation de mots-clés.

La méthode K.A.D.S. ([BREUCKE85]) se situe plus dans la catégorie des outils d'aide à la conception (phase 2 de la construction d'un SBC). Elle fournit un modèle de structuration de la connaissance. Celui-ci propose de séparer la connaissance en quatre catégories distinctes: on distingue en effet les connaissances de domaine (connaissances ontologiques sur les objets), les connaissances inférencielles (ensemble de contraintes ou d'actions conditionnelles sur les objets), les connaissances sur les tâches (connaissances sur la manière d'atteindre un objectif) et les connaissances liées à la stratégie d'exécution des tâches.
Comme pour K.O.D., la méthode K.A.D.S. a donné lieu à la réalisation d'une machine, l'atelier SHELLEY, dédiée à la conception de systèmes avec cette méthode. En termes de vérification, les fonctionnalités de cette machine se limitent à des facilités d'édition. Plus encore que pour K.O.D., on compte sur la formation du cogniticien à la méthode et sur la puissance d'outils style Hypertexte pour éviter les erreurs. Il n'y a donc pas de vérifications à proprement parler. K.A.D.S. pose seulement un certain nombre de bornes dans la conception du SBC, et incite le concepteur à utiliser des objets déjà validés (l'utilisateur dispose par exemple d'une bibliothèque de méthodes génériques dans laquelle il peut puiser les éléments nécessaires à la résolution de son problème).

d) L'approche Logique : Cohérence de Bases de Connaissances

d.1) Définition de la COHERENCE d'une Base de Règles

Enfin, quand les connaissances sont exprimées sous forme de règles (ie de formules de la forme $P \Rightarrow Q$), la vérification de bases de connaissances, peut être abordée en profitant, comme nous l'avons rappelé dans l'introduction, de leur lisibilité par la machine ([LAURIER87], p 356). C'est-à-dire du fait que la machine peut examiner et interpréter en terme de validité, le SBC qu'on lui a fourni.

De fait, les systèmes qui vont être présentés travaillent tous avec seulement la base de connaissances. Les vérifications consistent à combiner de (plus ou moins) toutes les façons possibles les connaissances contenus dans cette base, de manière à détecter:

- les cas de réponse absurde à un problème reconnu de la compétence du système,
- les bizarreries dans les règles, qui peuvent conduire à des déductions fausses ou incomplètes.

Concrètement, ces systèmes procèdent à un ou plusieurs des contrôles suivants:

- contrôle que les règles contradictoires ne peuvent conduire, lors d'un processus d'inférence à partir d'une base de faits initiale valide à la déduction de faits contradictoires.

Deux règles r: $P \Rightarrow Q$ et r': $P' \Rightarrow Q'$ sont contradictoires si Q contient q et Q' contient q' tel que:

. dans le modèle d'expression des connaissances utilisé, on sait que $q \wedge q' \Rightarrow \perp$

par exemple q désigne l'expression (E = 1), q' désigne l'expression (E = 2) et E est une entité monovaluée,

. on possède un ensemble de propriétés associées à la base de règles qui permettent de déduire que $q \wedge q' \Rightarrow \perp$.

Sauf dans le cas de SACCO (voir plus loin), ces propriétés seront toujours des contraintes de cohérence, c'est-à-dire des expressions de la forme $q \wedge q' \Rightarrow \perp$

par exemple q désigne l'expression SEXE_INDIVIDU = MALE, q' désigne l'expression CAUSE_ABSENCE_INDIVIDU = CONGE_MATERNITE, et on a la contrainte:

SEXE_INDIVIDU = MALE $\wedge$ CAUSE_ABSENCE_INDIVIDU = CONGE_MATERNITE $\Rightarrow \perp$

Les systèmes diffèrent les uns des autres par leur définition de la validité d'une base de faits initiale, par la prise en compte ou non de propriétés associées à la base de règles, et dans un cas, par une définition plus stricte des règles contradictoires.

Quand la base de règles satisfera ce premier contrôle, on dira qu'elle est COHERENTE.

- contrôle que la base de règles ne contient pas de règles:

. indéclenchables: r : $P \Rightarrow Q$ est indéclenchable si P n'est jamais démontrable lors d'un processus d'inférence à partir d'une base de faits initiale valide

. redondantes : r : $P \Rightarrow Q$ est redondante s'il existe r': $P' \Rightarrow Q'$ tq $P \supset P'$ et $Q' \supset Q$

. formant des cycles: un n-uplet de règles R1: $P1 \Rightarrow Q1$, ... , Rn: $Pn \Rightarrow Qn$ forme un cycle si $\forall i$ tq $1 \leq i < n$, $Qi \supset Pi+1$, et $Qn \supset P1$

Remarque: Les deux dernières propriétés ne concernent pas directement la fiabilité du système expert, sauf:

- s'il manipule des faits possédant des coefficients de vraisemblance: dans ce cas, la présence de règles redondantes peut entraîner de déduire trop de fois un même fait, faussant ainsi le coefficient de vraisemblance qui lui est associé.
- s'il raisonne en chaînage arrière ou mixte: dans ce cas, la présence de cycles peut entraîner des bouclages dans le processus d'inférence.

La présentation va séparer ces systèmes en deux catégories: les systèmes analysant les BC d'ordre Zéro Plus, et ceux analysant les BC d'ordre Un.

d.2) Les vérificateurs de BC d'ordre Zéro Plus

Rappelons qu'en logique d'ordre Zéro Plus,

- une prémisse est une expression de la forme: Entité Comparateur Valeur (ex. E > 2, A = 3),
- un conséquent, ou un fait, est une expression de la forme: Entité = Valeur
- qu'à tout instant, une entité donnée à au plus une valeur (monovaluation)

Les vérificateurs de bases de règles d'ordre Zéro Plus peuvent se diviser en deux groupes.

$\Rightarrow$ Le premier regroupe les outils se caractérisant par:

- But de l'analyse: recherche de la cohérence
- Critère de validité d'une base de faits initiale (BFi):

Une BFi est valide si elle ne contient que des entités non déductibles, et ne contient pas de couple de faits contradictoires (ie qui ne respectent pas la propriété de monovaluation des entités)

- Approche: exhaustive

On entend par là qu'ils mettent en oeuvre une méthode complète de preuve de la cohérence.

- Algorithme: on recherche les conditions minimales sur les entités non déductibles pour inférer les entités déductibles.
- les systèmes, leurs particularités:

INDE [PIPARD87] regarde également si les règles sont toutes déclenchables avec les hypothèses sur la BFi. il utilise un réseau de Pétri pour la représentation interne de la base de règles.

SUPER [FONT&al88] : Initialement orienté vers l'acquisition interactive et incrémentale de nouvelles règles, il en a gardé le coté interactif et incrémental.

COVADIS [ROUSSE88a] : A introduit le premier la notion de contraintes d'intégrité dans le contexte de la cohérence. On rajoute alors au critère de validité d'une BFi le fait que ces éléments doivent vérifier les contraintes de cohérence.

MELODIA [CHARLES90]: Travaille en logique booléenne, après transformation des BR dans ce formalisme. Tient également compte des contraintes d'incohérence associées à la base de règles vérifiée. Il est très performant (MELODIA a traité des bases de plusieurs milliers de règles) .

- Commentaire Global:

L'existence de ce groupe est le signe d'une véritable "école" de la vérification de la cohérence. Celle-ci a aussi, nous le verrons, des adeptes pour la vérification de BC d'ordre Un. En quelques mots, on pourrait la caractériser par: Objectifs restreints, mais atteints.

$\Rightarrow$ Le deuxième groupe contient ce que l'on serait tenter d'appeler les francs-tireurs. Ils proposent des vérifications plus larges et/ou tenant compte de la possibilité d'avoir des bases de faits initiales quelconques (à condition bien sur qu'elles ne contiennent pas de couple de faits contradictoires). On va les présenter successivement, en soulignant uniquement leur différence avec les caractéristiques énoncées ci-dessus:

KB-REDUCER [SUW&al82]:

- Buts de l'analyse: multiples, principalement:

. recherche des couples de règles contradictoires, en s'appuyant sur une définition spécifique: deux règles R1 et R2 sont contradictoires si les prémisses de l'une incluent les prémisses de l'autre et si leurs conclusions sont contradictoires.

. recherche de la complétude: détection de lacunes dans la BR quand un couple (entité déductible, valeur) est utilisé en prémisse sans être présent en conséquent, ou quand une combinaison d'hypothèses est négligé: par exemple si on a deux règles ayant comme prémisses respectivement $E = 1 \wedge F = 0$ et $E = 0 \wedge F = 1$, KB-REDUCER vérifie que la base contient deux règles ayant comme prémisses respectivement $E = 1 \wedge F = 1$ et $E = 0 \wedge F = 0$.

- Critère de validité d'une base de faits initiale (BFi): Une BFi est valide si elle ne contient pas de couple de faits contradictoires.

- Commentaire:

KB-REDUCER a choisi une définition de la cohérence moins stricte, et au fondement logique beaucoup moins fort. Clairement, la vérification de la cohérence est perçue ici comme une composante d'un système global de développement de SBC.

LRC3 [HERY85]:

- But de l'analyse: recherche de la cohérence statique définie par:

"BR est statiquement cohérente si quelque soit BR' inclus dans BR, quelque soit BF, l'ensemble de faits BR'.BF obtenu par saturation ne contient pas de couple de faits contradictoires".

Autrement dit, on ne veut pas ici s'intéresser à l'enchaînement des règles qui permet une déduction, on veut que "intrinsèquement", les règles soient non-contradictoires.

- Critère de validité d'une base de faits initiale (BFi): Une BFi est valide si elle ne contient pas de couple de faits contradictoires.

- Commentaire:

Un système qui a fait ses preuves, tout en s'appuyant sur une définition de la cohérence très contraignante pour l'utilisateur. Par contre, l'absence d'hypothèse restrictive sur la BFi en fait un garant précieux du bon fonctionnement du SBC en toutes circonstances.

JASMIN

- But(s) de l'analyse: recherche de la cohérence, en tenant compte des contraintes de cohérence associées à la base de règles étudiée

- Critère de validité d'une base de faits initiale (BFi): Une BFi est valide si elle ne contient pas de couple de faits contradictoires, et si ces éléments vérifient les contraintes de cohérence.

- Commentaire:

JASMIN résulte du pari un peu fou de réunir la souplesse de la définition de la cohérence du premier groupe d'outils avec la souplesse de la définition de la validité de la BFi du deuxième. Sur le papier, c'est l'alliance idéale. En pratique, le prix de cette souplesse et la génération d'un nombre impressionnant de contraintes que doit vérifier la BFi. L'utilisation de cet outil revient en effet presque à expliciter une à une toutes les BFi ne permettant pas de déduire de faits contradictoires.

d.3) Les vérificateurs de BC d'ordre Un

Rappelons que dans les bases de connaissances d'ordre Un,

- une prémisse ou un conséquent de règle est une expression de la forme: Attribut(Objet) Comparateur Valeur
avec Objet et Valeur des constantes ou des variables quantifiées universellement
(le Comparateur étant uniquement "="dans le cas d'un conséquent)
(ex PERE(X) = (Y), NUMERO(Z) > 5)

- un fait est un triplet Attribut(Objetconst) = Valconst avec Objetconst et Valconst deux constantes

Les outils de vérifications de BC d'ordre Un peuvent également se répartir en deux groupes.

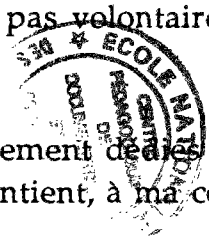
$\Rightarrow$ Le premier contient les outils construits comme des extensions à l'ordre Un d'outils ou de méthodes définis à l'ordre Zéro. On trouve dans ce groupe:

COCO-X [LOISEAU90] : cet outil est le prolongement à l'ordre Un de COVADIS. Il en hérite les mêmes restrictions sur les BFi et la même définition de la cohérence.

TIBRE [LALO89] : cet outil est également très imprégné des choix de COVADIS. Seule différence, l'étude des contradictions n'est pas exhaustive, mais guidée par l'utilisateur: c'est lui qui donne à TIBRE les contradictions qu'il doit étudier, ainsi que le niveau de précision qu'il doit adopter pour ces vérifications. Ces deux innovations sont guidées par la nécessité de traiter des cas réels, ce qui n'est pas le cas de COCO-X.

CHECK [NGU&al87] et ARC [NGUYEN87] : ce sont les outils pour les syntaxes LES et ART respectivement, reprenant le même schéma de contradiction et la même double vocation de vérification de cohérence et de complétude que KB-REDUCER. La filiation est d'ailleurs revendiquée.

TWAICE [MELL&R89]: C'est le prolongement à l'ordre Un de LRC3: même examen "statique", donc même degré de contrainte sur les règles, et même efficacité. La notion d'entité monovaluée est ici remplacée par la notion d'attribut monovalué. Remarquons que cette filiation n'est probablement pas volontaire, mais le résultat de la même démarche "industrielle".



⟹ Le deuxième groupe contient les travaux clairement dédiés à la logique et la sémantique des systèmes d'ordre Un. Ce groupe contient, à ma connaissance, deux membres:

- SACCO [AYEL87] devenu ultérieurement SACKOOL [TALB&A89] :
  - But(s) de l'analyse: Ce système opère la vérification de la conformité d'une base de connaissances (ici constitué d'un ensembles de règles et de faits) à son modèle.
  - Description du Modèle: Celui-ci est constitué d'un ensemble de propriétés sur les attributs présents dans la base de connaissances : valeurs autorisées et interdites, degré de valuation maximal, incompatibilité entre attributs, ...
  La vérification concerne toutes les facettes du modèle. Elle est donc multiforme.
  - Méthode d'Analyse: Toutes les vérifications possibles ne sont pas effectuées par SACCO, et la preuve d'une propriété donnée n'est pas non plus basée sur une exploration combinatoire de toutes les situations pouvant conduire au non-respect de cette propriété. Il s'agit ici de vérification heuristique. Plus précisément, c'est en se basant sur une troisième sorte de connaissance représentant le degré de confiance du concepteur sur une entité donnée (classes, attributs, objets, règles de déduction), elle aussi fournie par l'expert, que le choix des problèmes à étudier est effectué (on étudie bien sur en priorité les problèmes "sensibles")
  - Commentaire
  SACCO ne considère plus une base de connaissances comme un ensemble de formules logiques dont il faut vérifier la consistance, mais comme un ensemble de connaissance de natures diverses et de fonctions diverses (connaissances inférencielles, connaissances de domaine, connaissances pour utiliser les connaissances, connaissances pour acquérir les connaissances (terminologie empruntée au très précieux [PITRAT90])), dont il essaye de vérifier la cohérence. Il a montré la voie à la fois
    - de l'élargissement du concept de cohérence, en mettant en lumière le concept de modèle conceptuel d'une base de connaissance

- de retour à une démarche "plus IA", c'est-à-dire non plus algorithmique mais heuristique dans la vérification de la cohérence.

- le système de JL BARRIERE, esquissé dans [BARRIER90]:

Nous disposons à ce jour de très peu de renseignements sur ce travail. La seule publication disponible montre que ce système s'attaque au gros défaut des systèmes précédents: le refus de gérer la non-monotonie du processus d'inférence. Autrement dit, pour BARRIERE, il n'est pas contradictoire de nier certains faits au cours de l'inférence. Notre expérience nous fait partager cette opinion. Si l'hypothèse de monotonie de l'inférence est raisonnable pour des systèmes d'ordre zéro plus, elle ne l'est pas pour des systèmes d'ordre Un.

BARRIERE part des spécifications des bases de faits initiales possibles et procède à une simulation du comportement de la BC sur des BFi respectant ces spécifications. Pour échapper à l'explosion combinatoire (casi-immédiate!) de ce genre d'opérations, le système de BARRIERE se sert de métaconnaissances de d'exploitation de la BC, fournies par l'expert.

L'avenir dira si cette voie a pu aboutir, et à quel prix. Le constat de départ (la nécessité de gérer la non-monotonie) est, pour les bases de connaissances d'ordre Un, indiscutable, les idées sont sensées (utilisation d'un graphe d'état, recours massif à la Métaconnaissance), mais le problème tellement difficile ...

d.4) Métatravaux

Nous allons, pour terminer ce panorama, donner les références :

- de documents contenant d'autres états de l'art récents sur le même sujet. Ce sont: [AYE&al88], [ROUSSE88a], [LALO89], [MARTIN90], [LOPE&al90].

- de documents décrivant des formalisations logiques "strictes" du problème de la cohérence: [KLEI&al89], [ERMINE89]

2) Notre Démarche

a) Deux Logiques, Deux Systèmes

Nous avions pour but de vérifier des bases de connaissances écrites dans la syntaxe GENESIA1 (le langage basé sur la logique des propositions utilisé dans certaines applications de l'EDF) ou GENESIA2 (le langage basé sur la logique des prédicats utilisé dans de nombreuses applications à l'EDF). Nous allons sur un exemple justifier notre choix de réaliser deux systèmes distincts.

Prenons par exemple la proposition suivante écrite dans le langage GENESIA1: RCV.13VP.CONSIGNE = 25

Sémantiquement, cette proposition indique un test de comparaison (ou une affectation) entre la valeur de la pression au niveau d'une la vanne de décharge et (ou à) la valeur de référence qui est 25 bars. Examinons maintenant ce qu'il reste de cette sémantique dans la syntaxe, c'est-à-dire, essayons d'interpréter cette proposition en supposant que l'on ne connaisse pas ce que signifie RCV.13.CONSIGNE. Nous pouvons tout de même comprendre:

- qu'on est en train de regarder un test au sujet de la valeur d'un paramètre numérique
- que la valeur 25 est une valeur particulière pour cette entité

Mais surtout, le sens de cette proposition est sans ambiguïté:

- s'il s'agit d'une prémisse, elle signifie: "est-ce que le paramètre RCV... a atteint la valeur 25 ?",
- s'il s'agit d'un conséquent, elle signifie:: "dans les conditions énoncées en partie prémisse, la (nouvelle ?) valeur du paramètre RCV... vaut 25 ".

Bien sur, nous n'en déduirons pas ce que représente réellement l'entité, ni quelles sont les contraintes physiques associées à ce qu'elle représente, mais il est clair que le rôle de cette proposition dans la BR est lui, par contre, parfaitement explicite. Si on examine maintenant cette proposition dans une optique de vérification, les contrôles a effectuer viennent tout naturellement:

- Existe-t'il d'autres valeurs à laquelle est comparée, ou qui affectent cette entité? Sont-elles de type et de grandeur compatibles avec celle visible ici?
- S'il s'agit d'une affectation, la déduction d'autres valeurs est-elle possible? Si oui, est-ce dans un contexte exclusif avec celui de cette déduction?
- S'il s'agit d'un test, existe-t-il un conséquent permettant de satisfaire cette prémisse? Si non, existe-t'il un conséquent permettant d'affecter à l'entité RCV... une autre valeur ?

Les points importants sont donc:

- que les vérifications naturelles pour une base de règles d'ordre Zéro Plus tournent autour du problème de la **déduction de faits contradictoires**, et de la **satisfiabilité (déclenchabilité)** des règles qui la compose
- **que ces contrôles peuvent être définis, et conduits à partir de la base de règles seule.**

Celle-ci est vraiment lisible, car fondée sur un formalisme simple.

Reprenons maintenant l'expérience avec la proposition suivante écrite dans le langage GENESIA2 : A_p_N(X) = (Y)
Que pouvons-nous en dire ici , en l'absence de tout autre renseignement ? : Il existe une relation entre une entité quantifiée universellement (X) et une entité quantifiée universellement (Y). Et rien d'autre ! Quant à la vérification, elle pourrait à la rigueur consister à essayer de chercher d'autres propositions de la forme non(A_p_N(Z) = (T)) et examiner leur rapport, mais GENESIA2 ne contient de possibilité d'exprimer la négation[3]!

Nous ne pouvons donc rien analyser. En effet, si les systèmes écrits dans cette syntaxe ont gardé la modularité dans l'expression de la connaissance, ils ont perdu la majeure partie de leur lisibilité. **Pour les rendre à nouveau accessibles à un regard "machine", il faut donner à celle-ci un ensemble de connaissances supplémentaires.**

Par exemple, sur la proposition précédente, imaginons que l'on sache que A_p_N est une relation de Nommage (A_p_N est la forme abrégé de A_pour_Nom). Alors seulement, on peut déduire (et vérifier):

- que (X) représente une entité complexe à laquelle on a besoin d'associer un identificateur et que (Y) est le nom de (X), c'est-à-dire représente un objet élémentaire du type "une chaîne de caractères"

[on peut vérifier que ces informations ne sont pas contradictoires avec les autres occurrences de (X) et (Y) dans la règle: par exemple, une prémisse de la forme (Y) > (Z) serait surprenante]

- que (Y) est certainement le seul nom de (X), voire peut-être que (X) est la seule entité de nom (Y),

[ie que la relation A_p_N est une fonction, voire une bijection: on pense alors à des vérifications comparables à celles pratiquées en logique zéro plus, du type : "N'y a-t-il pas un risque pour une entité d'avoir deux noms?"]

- que ce prédicat peut être utilisé [doit être utilisé ?] pour s'assurer que deux objets du type de ceux que représente X et instanciant deux variables X1 et X2 sont différents, grâce à des prémisses respectant le schéma:
A_p_N(X1) = (Y1) et A_p_N(X2) = (Y2) et (Y1) ≠ (Y2)

- en allant plus loin, que la relation A_p_N résulte certainement de la décomposition d'une relation n-aire en n relations binaires (GENESIA 2 n'admet que les relations binaires), donc que (X) représente le pivot de cette opération et (Y) le nom de ce qui à été décomposé: par exemple, (X) représente le radical de décomposition d'une tache : T1, T2, ... ; et (Y) le nom réel de la tache : NETTOYAGE, ENDUCTION, POLISSAGE, MISE-EN-PEINTURE, ...; cette tache ayant par ailleurs une durée, une date de début au plus tôt, une date de début au plus tard, etc ...

[on peut donc chercher quelles sont les relations binaires liées à A_p_N, et vérifier qu'à toute création d'une instance de cette relation correspond la création d'une instance pour chacune des relations "soeurs"]

Cette liste n'est pas exhaustive. Elle montre que la possession d'informations de ce type est essentielle.

---

[3] sauf dans un cas assez marginal, et peu employé.

Les points importants sont donc ici:

- que les vérifications envisageables pour une base de règles d'ordre Un sont très variées, et peuvent correspondre à des problèmes soit de faits "mal-formés", soit de contradiction entre règles, soit de liens de déclenchements entre règles, etc ...

- **que ces contrôles ne peuvent être définis, et conduits qu'à partir du moment où l'on possède des connaissances supplémentaires sur la modélisation employée pour construire la base de règles .**

La problématique est donc différente dans le cas de vérifications de base de règles d'ordre Zéro Plus ou d'ordre Un. Les types de vérifications, ainsi que les connaissances nécessaires pour les effectuer sont de nature casi-distinctes. **Nous avons donc conçu deux systèmes, un pour chacune de ses logiques.** Nous allons présenter plus précisément leurs finalités.

b) Le Système de Vérification de Base de Règles d'ordre Zéro Plus

Nous avons déjà annoncé que nous nous occuperons principalement dans ce contexte de la **vérification de la non-contradiction entre règles, le concept de contradiction reposant, pour la logique zéro plus, sur le fait qu'une entité ne peut être multivaluée.**
Nous le ferons avec les mêmes hypothèses simplificatrices que celles décrites dans la partie 1)d)d.2 ("école" COVADIS, MELODIA etc ...).
Néanmoins, nous avons constaté que cette vérification engendrait ou rendait possible d'autres traitements. Nous avons donc associé au programme de vérification de la non-contradiction entre règles d'autres modules, dits de "gestion de l'amont" et de "gestion de l'aval" de ce traitement.

En quelques mots, gérer l'amont (de l'inférence) signifiera:

- **épurer la base de règles de ces règles inutiles (ce qui recouvre le contrôle de la présence de règles indéclenchables et de règles redondantes déjà évoqué)**

- constituer un filtre détectant les bases de faits initiales susceptibles d'engendrer des contradictions à l'inférence

tandis que gérer l'aval signifiera: mettre à la disposition de l'utilisateur un outil de visualisation des règles impliquées dans une démonstration contradictoire.

c) Le Système d'Aide au Développement de Bases de Connaissances d'ordre Un

Nous avons montré que la possession d'informations sur les prédicats présents dans la base de règles était indispensable pour mener n'importe quelle tâche de vérification. Notre premier travail a donc été de définir un cadre permettant de fournir cette nouvelle connaissance. Nous avons ainsi défini **un modèle de description des prédicats,** qui permet à l'utilisateur de signaler à la machine les propriétés des prédicats de la base de règles étudiée.
Un des aspects de notre système sera donc, comme dans SACCO, de confronter la base de règles avec les propriétés exprimées grâce au modèle de description des prédicats. On aura donc une composante "**Vérification de la cohérence d'une base de connaissances**".

Mais, nous avons également pensé **que cette nouvelle connaissance devait être intéressante en tant que telle,** c'est-à-dire en dehors du contexte de vérification de la BR. Aussi avons-nous conçu un modèle de description des prédicats qui permette de fournir des connaissances que la machine n'est pas capable pour l'instant d'exploiter ou de vérifier, **mais qui sont fondamentales pour comprendre le "comportement" de la base de règles.**
Poursuivant cette logique, nous avons voulu faire suivre certaines phase de vérification par **des phases de découverte de nouvelles propriétés,** sorte d'**inversion des rôles,** en faisant de la base de règles l'outil de développement de la base contenant les propriétés des prédicats.
En cela, notre système est devenu un outil d'**aide à la spécification** de systèmes experts basés sur des règles d'ordre Un.

Enfin, notre système avons voulu essayer, en marge du travail évoqué ci-dessus:

- qu'une partie du travail de notre système soit d'être simplement **loquace,** c'est-à-dire consiste à **reformuler et regrouper** les informations fournies par la base de règles ou l'ensemble de propriétés. A charge alors à l'utilisateur d'interpréter cette nouvelle formulation.
- que notre système teste la conformité des règles et des bases de règles à des heuristiques de bon sens et signale "**les bizarreries syntaxiques**" ainsi détectées. Ces heuristiques reflètent nos "tics" de programmation. Cette vérification est donc totalement empirique, et aborde un autre aspect de la notion de **Qualité** d'une base de règles.

PARTIE 1: Cohérence de Bases de Règles d'Ordre Zéro Plus

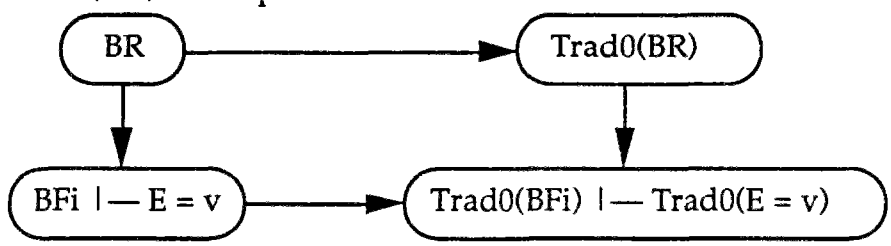
CHAPITRE II: Traduction, Définitions des Objectifs

1) Introduction

Le but de ce chapitre est double. D'une part, il s'agit de démontrer la validité des opérations transformant une base de règles écrite dans la syntaxe GENESIA1_TR en un système formel basé sur celui de la logique des propositions. D'autre part, on définira ensuite l'équivalent dans ce système formel de la propriété de Cohérence d'une base de règles.

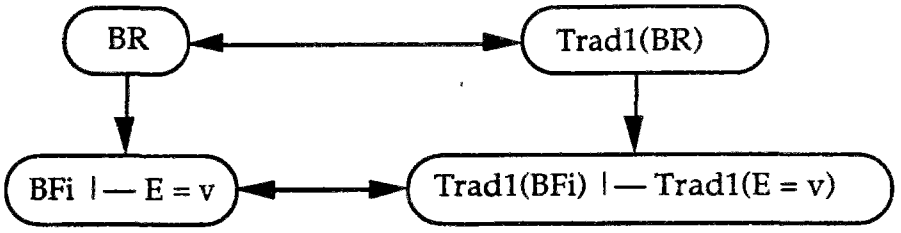
Nous n'allons pas toutefois justifier l'intégralité de la transformation d'une base de règles en un système formel. En effet, notre travail se situe dans le prolongement d'une démarche qui a déja été démontrée valide. Nous ne justifierons donc que ce qui nous appartient en propre. Commençons donc par situer nos travaux par rapport à leurs prédécesseurs.

A) [CHARLES90] a montré comment convertir une base de règles d'ordre Zéro Plus en un ensemble de contraintes de la forme: $\wedge a_i \Rightarrow c$, avec a_i et c des littéraux (ie des variables propositionnelles ou leur négation), "$\wedge$" le ET et "$\Rightarrow$" l'implication de la logique des propositions.
Cette traduction a été montrée complète dans [DUB&al90].
On a donc le schéma classique suivant: à une base de règles BR correspond un ensemble de contraintes Trad0(BR), et à toute expression démontrable dans BR à partir d'une base de faits initiale BFi - ie à toute expression entité = valeur, notée E = v, correspond une démonstration de l'équivalent de E = v dans le nouveau système (noté Trad0(E = v)), ayant pour hypothèses l'ensemble de contraintes Trad0(BFi) correspondant à BFi :



On notera le caractère orienté des relations. Il signifie:
- pour la flèche du haut: qu'à partir du système Trad0(BR), on ne peut pas revenir à BR
- pour la flèche du bas: que s'il est vrai qu'à toute inférence à partir d'une base de faits initiale (valide) correspond une démonstration dans le système traduit, la réciproque est fausse. Autrement dit, des démonstrations dans le système traduit n'ont pas d'équivalent dans le système initial.

B) Nous inspirant de ces travaux, nous avons établi un deuxième modèle de traduction, extrèmement proche de celui dont il est question ci-dessus mais qui modèlise plus finement le comportement de la base de connaissances.
On a en effet avec le deuxième système, en reprenant les mêmes notations que ci-dessus et en appelant Trad1 ce deuxième opérateurs de traduction les ralations suivantes:



Ce qui signifie:

- que cette fois la traduction est réversible: on peut regénérer le système initial à partir du système traduit
- qu'il y a équivalence entre les "pouvoirs de déduction" des deux systèmes.

Dans ce nouveau modèle, une base de règles est transformée en:

- un ensemble de productions, ie d'expressions de la forme: $\wedge a_i \rightarrow c$

avec a_i et c des variables propositionnelles, une production étant exploitable dans le système formel (SF en abrégé) par la seule règle (E→): $\dfrac{A \qquad A \rightarrow B}{B}$

- d'un ensemble de contraintes logiques, ie d'expressions de la forme

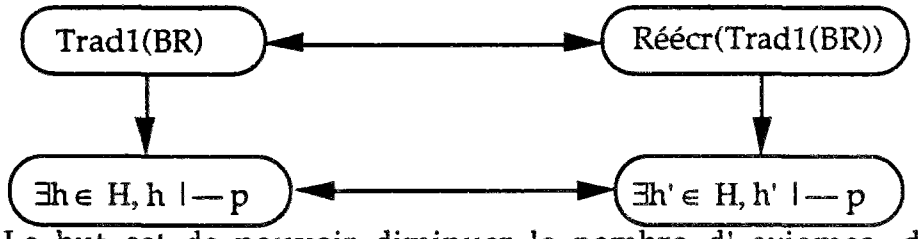
$p \Rightarrow q$ ou $p \Rightarrow \neg q$ ou $\neg p \Rightarrow q$ avec p et q des variables propositionnelles et "⇒" l'implication logique, une contrainte logique étant exploitable dans le système formel (SF en abrégé) par les deux règles de déduction:

$(E\Rightarrow_1)$: $\dfrac{p \qquad p \Rightarrow q}{q}$ et $(E\Rightarrow_2)$: $\dfrac{\neg q \qquad p \Rightarrow q}{\neg p}$

La différence entre Trad1 et Trad0 est néanmoins très petite, comme nous le verrons immédiatement après sur l'exemple. Cette deuxième traduction profite donc des propriétés démontrées dans [DUB&al90]. Elle est complète.

C) Nous allons pour notre part montrer dans ce chapitre **un algorithme de réécriture des contraintes logiques issues de la traduction évoquée ci-dessus qui permet,** à l'issue de l'opération:

1°) de **confondre contraintes et productions,** tout en conservant, pour une classe d'hypothèses initiale H précisée, les mêmes capacités déductives. On aura ici les relations :



Le but est de pouvoir diminuer le nombre d' axiomes, donc de simplifier les vérifications concernant les déductions possibles.

2°) de **minimiser le nombre de contraintes,** en tirant partie du fait qu'avec les règles de déduction de notre système formel et la restriction sur les hypothèses initiales valides, certaines contraintes sont inexploitables.

3°) de **conserver la propriété fondamentale d'un fait initial valide pour l'inférence** avec BR (ce qui est connu), **pour l'équivalent de ce fait initial** dans le système formel (SF en abrégé) issu de la réécriture, à savoir:
pour BR: "un fait initial valide est un couple (E, val) avec E une entité désignée initiale par l'expert et **n'apparaissant jamais en partie droite d'une règle de BR**, et val une valeur"
pour le SF issu de la traduction: "L' (ou les) hypothèse(s) initiale(s) valide(s) correspondant à un fait initial valide est (sont) un (des) littéral (littéraux) **n'apparaissant jamais en partie droite d'une production ou d'une contrainte**".

Cette propriété n'était pas conservée avant réécriture. Cela rendait les définitions et les démonstrations (entr'autres celle de l'équivalence des deux systèmes sur le plan de la cohérence) plus délicates.

2) Exemple

Illustrons ce cheminement sur un exemple.
Soit la base de règles:

reg1:	Si	EI1 = 5	Alors	E = 1			
reg2:	Si	EI1 > 0	Alors	E = 2			
reg3:	Si	E < > 1	Alors	F = 1			
reg4:	Si	E < > 2	Alors	F = 2			
reg5:	Si	E I2 = 'vrai'	Et	EI1 > 0	Alors	A = 1	
reg6:	Si	A > 0	Alors	B = 1			
reg7:	Si	B > 0	Alors	E = 3			
reg8:	Si	B < 0	Alors	F = 2			

A) la première traduction (Trad0, cf [CHARLES90]) consiste à réécrire le système sous la forme:

1) contraintes issues directement des règles

$m1 \Rightarrow m3$
$m2 \Rightarrow m4$
$\neg m3 \Rightarrow m5$
$\neg m4 \Rightarrow m6$
$m7 \wedge m2 \Rightarrow m8$
$m9 \Rightarrow m10$
$m11 \Rightarrow m12$
$m13 \Rightarrow m6$

avec

m1	représentant	EI1 = 5
m2	représentant	EI1 > 0
m3	représentant	E = 1
m4	représentant	E = 2
m5	représentant	F = 1
m6	représentant	F = 2
m7	représentant	EI2 = 'vrai'
m8	représentant	A = 1
m9	représentant	A > 0
m10	représentant	B = 1
m11	représentant	B > 0
m12	représentant	E = 3
m13	représentant	B < 0

2) contraintes logiques entre variables propositionnelles

m1 ⇒ m2	correspondant à:	EI1 = 5 ⇒ EI1 > 0
m3 ⇒ ¬m4	" "	E = 1 ⇒ E <> 2
m3 ⇒ ¬m12	" "	E = 1 ⇒ E <> 3
m4 ⇒ ¬m12	" "	E = 2 ⇒ E <> 3
m5 ⇒ ¬m6	" "	F = 1 ⇒ F <> 2
m8 ⇒ m9	" "	A = 1 ⇒ A > 0
m10 ⇒ m11	" "	B = 1 ⇒ B > 0
m10 ⇒ ¬m13	" "	B = 1 ⇒ ¬(B < 0)
m11 ⇒ ¬m13	" "	B > 0 ⇒ ¬(B < 0)

B) Notre première traduction est identique à celle ci-dessus, sauf pour les contraintes issues directement des règles pour lesquelles on remplace "⇒ "par "→", pour tenir compte du fait que les règles ne sont exploitées que par le Modus Ponens, donc ne doivent pas être utilisées sous forme contraposée.

C) La réécriture s'effectue en trois étapes:

1°) **Duplication** de toutes les contraintes logiques p ⇒ q avec p et q des littéraux en $p \rightarrow q$ et $\neg q^1 \rightarrow \neg p^2$ (pour pouvoir supprimer ($E \Rightarrow_2$))

2°) a) **Suppression des contraintes inutiles pour les déductions**

Si p ne peut être démontré (ie n'est pas une hypothèse initiale valide et n'est pas présent en conséquent d'au moins une production)

OU si q n'est pas utile dans les démonstrations (il n'apparait dans aucune partie prémisse de production)

Alors Supprimer la contrainte $p \rightarrow q$

b) **Conservation de l'exclusion entre variables**

Si les deux contraintes $p \rightarrow q$ et $\neg q \rightarrow \neg p$ ont été supprimées par l'étape a), écrire la contrainte $p \wedge \neg q \rightarrow \perp$ [3].

Cette réécriture donne ici (on donne à gauche la contrainte initiale et à droite la (les) nouvelle(s) contrainte(s) après réécriture et son (leurs) équivalent(s) "en clair"):

m1 ⇒ m2 ——————————> $m1 \wedge \neg m2 \rightarrow \perp$ (EI1 = 5 et EI1 ≤ 0 → ⊥)

duplication
suppression
conservation de l'exclusion

m2 et ¬m1 sont des hypothèses initiales, elles n'apparaissent pas en partie droite de production

---

[1] Cette écriture suppose implicitement que ¬¬p = p. En fait, le SF ne déduit jamais de terme de la forme ¬¬p, donc la question ne se pose pas. Il ne s'agit ici que d'une façilité d'écriture, pour éviter d'écrire Opposé(q) avec Opposé la fonction définie sur l'ensemble des littéraux par:
si l est un littéral et que l = x avec x une variable propositionnelle, Opposé(l) = ¬x
si l est un littéral et que l =¬ x avec x une variable propositionnelle, Opposé(l) = x

[2] idem ci-dessus

[3] Nous verrons que cette deuxième étape a pour conséquence que si p (ou ¬p) ou q (ou ¬q) est une hypothèse initiale valide, alors une contrainte du type p ⇒ q est réécrite $p \wedge \neg q \rightarrow \perp$, ce qui réalise l'objectif de faire disparaitre les hypothèses initiales des parties conséquents des contraintes.

$m3 \Rightarrow \neg m4$	——————————> duplication	$m3 \rightarrow \neg m4$ $(E = 1 \rightarrow E <> 2)$ $m4 \rightarrow \neg m3$ $(E = 2 \rightarrow E <> 1)$
$m3 \Rightarrow \neg m12$	——————————> duplication, suppression	$m12 \rightarrow \neg m3$ $(E = 3 \rightarrow E <> 1)$ $\neg m12$ n'apparait pas en prémisse
$m4 \Rightarrow \neg m12$	——————————> idem ci-dessus	$m12 \rightarrow \neg m4$ $(E = 3 \rightarrow E <> 2)$
$m5 \Rightarrow \neg m6$	——————————> duplication, suppression conservation de l'exclusion	$m5 \wedge m6 \rightarrow \perp$ $(F = 1$ et $F = 2 \rightarrow \perp)$ ni $\neg m5$, ni $\neg m6$ n'apparaissent en prémisse de production
$m8 \Rightarrow m9$	——————————> duplication, suppression	$m8 \rightarrow m9$ $(A = 1 \rightarrow A > 0)$ $\neg m8$ n'apparait pas en prémisse
$m10 \Rightarrow m11$	——————————> idem cas précédent	$m10 \rightarrow m11$ $(B = 1 \rightarrow B > 0)$
$m10 \Rightarrow \neg m13$	——————————> duplication, suppression conservation de l'exclusion	$m10 \wedge m13 \rightarrow \perp$ $(B = 1$ et $B < 0 \rightarrow \perp)$ ni $\neg m10$, ni $\neg m13$ n'apparaissent en prémisse de production
$m11 \Rightarrow \neg m13$	——————————> idem ci-dessus	$m11 \wedge m13 \rightarrow \perp$ $(B > 0$ et $B < 0 \rightarrow \perp)$

3) Cadre: Présentation et adaptation

On va maintenant décrire formellement l'opération de réécriture des contraintes logiques. Les notations définies dans cette partie seront aussi valables pour le chapitre suivant.

a) Les Systèmes Formels concernés

a.1) Présentation

On part de Systèmes Formels issus d'une traduction (Trad1) identique à celle exposée dans [CHARLES90], mais dans laquelle on distingue, dans l'ensemble des axiomes du SF, les productions (formules issues de la traduction directe des règles de la BR de départ) des contraintes logiques (formules exprimant les liens entre mots représentant les différentes formes d'occurrence d'une même entité).
[-> Pour la description de l'obtention de ce SF, voir le chapitre 4, pragraphe 2). Pour la preuve de la validité de l'opération, voir [DUB&al90]]

Ces systèmes formels sont de la forme:

1) Vocabulaire: Voc = {$\neg$, $\wedge$, $\rightarrow$, $\Rightarrow$, les mots courants}

2) Règles de Formation des phrases: $\mathcal{RF}$
- si m est un mot courant, m et $\neg$m sont des phrases appelées littéraux,
- si p et q sont des littéraux ou des prémisses, p, q et p $\wedge$ q sont des prémisses
- si p est une prémisse et q un mot courant, p $\rightarrow$ q est une production

- si p et q sont des littéraux, $p \Rightarrow q$ est une contrainte logique

3) Les Axiomes sont de deux types:
a1- un ensemble BR de productions
a2- un ensemble EC de contraintes

4) Les Règles de Déduction sont: $\mathcal{RD}$

$(I\wedge)$: $\dfrac{A \quad B}{A \wedge B}$ $\qquad$ $(E\rightarrow)$: $\dfrac{A \quad A \rightarrow B}{B}$

$(E\Rightarrow_1)$: $\dfrac{p \quad p \Rightarrow q}{q}$ $\qquad$ $(E\Rightarrow_2)$: $\dfrac{\neg q \quad p \Rightarrow q}{\neg p}$

Un tel système est caractérisé par l'ensemble {Voc, $\mathcal{RF}$, BR ∪ EC, $\mathcal{RD}$}.
On écrira SF = {Voc, $\mathcal{RF}$, BR ∪ EC, $\mathcal{RD}$}.

a.2) Définitions et Notations

* On appelera **SFBC** un SF du type décrit ci-dessus et dont les axiomes sont uniquement des productions ou des contraintes (**BR** désigne l'ensemble des productions, **EC** désigne l'ensemble des contraintes et **BC** l'union de ces deux ensembles)

* On confondra souvent l'expression H: $A \wedge B \wedge C \wedge ...$
avec l'ensemble H = {A, B, C, ...}, et réciproquement.
de plus, on notera "-" l'opérateur "**moins ensembliste**". Les éléments de E - F seront donc les éléments de E n'appartenant pas à F.

* On note **Abs** la fonction de l'ensemble des littéraux dans l'ensemble des mots défini par: Abs(m) = m et Abs(¬m) = m
Par extension, on notera aussi Abs la fonction de l'ensemble des prémisses dans lui-même défini par: $Abs(A \wedge B) = Abs(A) \wedge Abs(B)$,
On note **Opposé** la fonction des parties de l'ensemble des littéraux dans elles-mêmes telle que: $Opposé(H) = \{\neg h, h \in H\} \cup \{h, \neg h \in H\}$

* Ensuite, on définit, pour E = ER ou E = EC:
Cons(E) l'ensemble des littéraux apparaissant en conséquent des éléments de E,
Prem(E) l'ensemble des littéraux apparaissant en prémisse des éléments de E.
MO_E l'ensemble des mots apparaissant dans E: $MO_E = Abs(Prem(E)) \cup Abs(Cons(E))$

* Enfin, on définit les ensembles de mots suivants:
MD_{BR} (mots déductibles dans BR) l'ensemble des mots apparaissant en partie droite des éléments de BR: $MD_{BR} = Abs(Cons(BR))$
MD_{EC} l'ensemble des mots apparaissant dans une contrainte de EC en compagnie d'un élément de MD_{BR} et n'appartenant pas déjà à MD_{BR}: $MD_{EC} = MO_{EC} - MD_{BR}$

MD_{BC} l'union de MD_{BR} et MD_{EC} : $MD_{BC} = MD_{BR} \cup MD_{EC}$
MNR_{BC} (mots non récepteurs) l'ensemble complémentaire de MD_{BC} dans MO_{BC} :
$MNR_{BC} = MO_{BC} - MD_{BC}$

Nous ne nous intéresserons toujours seulement qu'aux démonstrations effectuées à partir d'hypothèses initiales construites à partir de mots de MNR_{BR} (cette définition sera précisée dans le 4).
Une hypothèse initiale valide est donc un littéral h tel que Abs(h) $\in MNR_{BC}$.

* Pour finir, une production de la forme r: $\wedge l_i \rightarrow c$ avec l_i et c des littéraux sera notée sous la forme **r: Prem(r) → Cons(r).**
Prem(r) désigne la partie prémisse de la production et Cons(r) la partie conséquent.
On dira que **$l_i \in$ Prem(r)** (resp. $\notin$ Prem(r)) pour signifier que l_i est (resp. n'est pas) un des littéraux de la conjonction Prem(r), et **c $\in$ Cons(r)** pour c = Cons(r).

a.3) Règles sur les contraintes logiques

Les contraintes logiques ne sont pas quelconques (voir chapitre 4) 2) c)). Outre le fait qu'elles soient de la forme p ⇒ q avec p et q des littéraux, elles obéissent à des lois.
Notons **$\mathcal{R}$(EC, ⇒, p, q)** la relation (symétrique) "Il existe dans EC une contrainte soit entre Abs(p) ou ¬Abs(p) d'une part, et Abs(q) ou ¬Abs(q) d'autre part, soit entre Abs(q) ou ¬Abs(q) et Abs(p) ou ¬Abs(p)", et **EC(⇒, p, q)** la relation "Il existe dans EC la contrainte logique p ⇒ q ou la contrainte ¬q ⇒ ¬p"

Les règles sur les contraintes sont les suivantes:

l1) **Transitivité de $\mathcal{R}$ et de EC:**
Si on a $\mathcal{R}$(EC, ⇒, p, q) et $\mathcal{R}$(EC, ⇒, q, r), alors on a forcément $\mathcal{R}$(EC, ⇒, q, r)
Si on a EC(⇒, p, q) et EC(⇒, q, r), alors on a EC(⇒, q, r)
La première règle signifie que sur chacun des ensemble de mots reliés entre eux par "⇒", ($\mathcal{E}$ = {Abs(o) / ∃o' tq o ⇒ o' ou o' ⇒ o), la relation "⇒" forme une clique (tous les objets sont reliés deux à deux par "⇒").
La deuxième montre que l'ensemble EC respecte la transitivité de l'implication logique en mathématique.

l2) **Aspect Déductif**
Si on a EC(⇒, p, q), alors
l2.1) p $\notin$ Cons(BR) ou q $\notin$ Cons(BR)
l2.2) ¬p $\notin$ Cons(BR) ou ¬q $\notin$ Cons(BR)
Remarque: La relation "n'appartient pas à Cons(BR)" doit ici s'interpréter comme "appartient à Prem(BR)" ou "n'apparait pas dans les productions".

b) Réécriture des contraintes sous forme de productions

b.1) Introduction de l'atome de contradiction, Algorithme de réécriture

Pour simplifier la vérification de la cohérence, nous allons réécrire les contraintes logiques sous forme de productions. La différence entre ces deux formules logiques étant l'existence, dans le cas des contraintes logiques, d'une utilisation possible sous forme contraposée, la solution triviale consisterait à remplacer chaque contrainte $p \Rightarrow q$ par les deux productions $p \rightarrow q$ et $\neg q \rightarrow \neg p$. Néanmoins, pour tenir compte de la nature particulière des contraintes que nous manipulons (cf paragraphe précédent), nous avons adopté d'autres règles de réécriture que nous allons maintenant détailler.

Celles-ci nécessitent l'introduction d'un nouveau symbole, **noté** $\perp$, et appelé **atome de contradiction** ou **symbole de l'absurde**. Il ne sera présent qu'en conséquent de production. Sa sémantique s'interprète de la manière suivante:
"$A \rightarrow \perp$" signifie "A ne peut pas être vrai"[4].
Enfin, on étend Abs à $\perp$ de la manière suivante: **Abs($\perp$) = $\perp$**.

L'algorithme repose sur l'hypothèse que les contraintes logiques sont stockées dans une liste. Il construit l'ensemble, noté EC', des nouvelles contraintes logiques.

Algorithme "Réécriture des Contraintes Logiques du SF = {Voc, $\mathcal{RF}$, BR $\cup$ EC, $\mathcal{RD}$}"
EC' $\Leftarrow \varnothing$, m $\Leftarrow$ Premier_Elément_De(EC) TANT QUE (m <> NIL) écrire_absurde $\Leftarrow$ Vrai, p $\Leftarrow$ Partie_Gauche_De(m), q $\Leftarrow$ Partie_Droite_De(m) (* Règle 1 *) SI (p $\in$ Cons(BR) et q $\in$ Prem(BR)) ALORS écrire_absurde $\Leftarrow$ Faux, EC' $\Leftarrow$ EC' $\cup$ {p $\rightarrow$ q} (* Règle 2 *) SI ($\neg$p $\in$ Prem(BR) et $\neg$q $\in$ Cons(BR)) ALORS écrire_absurde $\Leftarrow$ Faux, EC' $\Leftarrow$ EC' $\cup$ {$\neg$q $\rightarrow$ $\neg$p} (* Règle 3 *) SI écrire_absurde = Vrai ALORS EC' $\Leftarrow$ EC' $\cup$ {p $\wedge$ $\neg$q $\rightarrow$ $\perp$} m $\Leftarrow$ Elément_Suivant_De(EC) FIN_TANT_QUE

[4] La sémantique que nous associons à "$\perp$" est moins forte que celle qui lui est attribué classiquement, à savoir: $A \rightarrow \perp$ = "A est faux", (cf [PABION89]). La principale différence est que:
- avec la sémantique classique si on a à la fois $A \wedge B \rightarrow \perp$ et A, alors on a forcément $\neg B$ et de même si on a B, on a $\neg A$.
- alors qu'avec notre sémantique, on a seulement "B n'est pas vrai" ou "A n'est pas vraie".
Dans les cas où A doit vraiment entraîner $\neg B$ (par exemple), on écrira explicitement $A \rightarrow \neg B$.

Remarques:

1°) Les deux premières règles peuvent être satisfaites pour une même contrainte. Ceci correspond au cas ou l'on duplique la contrainte $p \Rightarrow q$ en $p \rightarrow q$ et $\neg q \rightarrow \neg p$.

2°) On peut réécrire les règles 1 et 2 de la manière suivante:

Si EC($\Rightarrow$, p, q) Alors
si $p \in$ Cons(BR) et $q \in$ Prem(BR)
alors écrire $p \rightarrow q$
si $\neg p \in$ Prem(BR) et $\neg q \in$ Cons(BR)
alors écrire $\neg q \rightarrow \neg p$

Nota Bene:

Pour pouvoir continuer à distinguer dans les démonstrations de ce chapitre les productions issues des contraintes des productions issues directement des règles, tout en les considérant toutes de la même façon, **on maintient le connecteur "$\Rightarrow$" dans les contraintes réécrites, et on supprime la règle de déduction ($E\Rightarrow_2$) du nouveau système formel obtenu.**

Donc, si le système formel initial était caractérisé par le quadruplet:
SF1 = {Voc, $\mathcal{RF}$, BR $\cup$ EC, $\mathcal{RD}$}
Celui après réécriture sera caractérisé par:
SF2 = {Voc $\cup$ {$\perp$}, $\mathcal{RF}$, BR $\cup$ EC', $\mathcal{RD} - \{(E\Rightarrow_2)\}$}.

A partir du chapitre suivant, comme nous ne considèrerons plus que des systèmes réécrits, on n'emploiera plus que le symbole "$\rightarrow$".

b.2) Exemple

On reprend l'exemple du paragraphe 2), avec les nouvelles notations.
Soit le système formel issu de la traduction suivant:
SF1 = {{¬, ∧, ⇒, →, MO_{BC}}, $\mathcal{RF}$, BR ∪ EC, $\mathcal{RD}$},
avec BR l'ensemble des productions issues des règles:

m1 → m3	m7 ∧ m2 → m8
m2 → m4	m9 → m10
¬m3 → m5	m11 → m12
¬m4 → m6	m13 → m6

MO_{BC} l'ensemble des mots du SF:

m1	(EI1 = 5)	m4	(E = 2)	m7	(EI2 = 'vrai')	m10	(B = 1)
m2	(EI1 > 0)	m5	(F = 1)	m8	(A = 1)	m11	(B > 0)
m3	(E = 1)	m6	(F = 2)	m9	(A > 0)	m12	(E = 3)
						m13	(B < 0)

EC l'ensemble des contraintes logiques:

m1 ⇒ m2	(EI1 = 5 ⇒ EI1 > 0)	m5 ⇒ ¬m6	(F = 1 ⇒ F <> 2)
m3 ⇒ ¬m4	(E = 1 ⇒ E <> 2)	m8 ⇒ m9	(A = 1 ⇒ A > 0)
m3 ⇒ ¬m12	(E = 1 ⇒ E <> 3)	m10 ⇒ m11	(B = 1⇒ B > 0)
m4 ⇒ ¬m12	(E = 2 ⇒ E <> 3)	m10 ⇒ ¬m13	(B = 1⇒ ¬(B < 0))
		m11 ⇒ ¬m13	(B > 0 ⇒ ¬(B < 0))

D'où : Cons(BR) = {m3, m4, m5, m6, m8, m10, m12}
Prem(BR) = {m1, m2, ¬m3, ¬m4, m7, m9, m11, m13}
MD_{BR} = {m3, m4, m5, m6, m8, m10, m12}
MD_{EC}= {m9, m11, m13}
MNR_{BC} = {m1, m2, m7}

On a alors EC', nouvel ensemble de contraintes logiques (écrites avec ⇒, cf Nota Bene):

m1 ∧ ¬m2 ⇒ ⊥	(EI1 = 5 et EI1 > 0 ⇒ ⊥) :	réécriture avec la P ègle 3
m3 ⇒ ¬m4	(E = 1 ⇒ E <> 2) :	" " " " " 1
m4 ⇒ ¬m3	(E = 2 ⇒ E <> 1) :	" " " " " 2
m12 ⇒ ¬m3	(E = 3 ⇒ E <> 1) :	" " " " " 2
m12 ⇒ ¬m4	(E = 3 ⇒ E <> 2) :	" " " " " 2
m5 ∧ m6 ⇒ ⊥	(F = 1 et F = 2 ⇒ ⊥) :	" " " " " 3
m8 ⇒ m9	(A = 1 ⇒ A > 0) :	" " " " " 1
m10 ⇒ m11	(B = 1 ⇒ B > 0) :	" " " " " 1
m10 ∧ m13 ⇒ ⊥	(B = 1 et B < 0 ⇒ ⊥) :	" " " " " 3
m11 ∧ m13 ⇒ ⊥	(B > 0 et B < 0 ⇒ ⊥) :	" " " " " 3

et SF2 = {Voc ∪ {⊥}, $\mathcal{RF}$, BR ∪ EC', $\mathcal{RD}$ – {($E\Rightarrow_2$)}}

REMARQUE: **Si $P \Rightarrow Q \in EC'$, avec $Q \neq \perp$, alors $P \in Cons(BR)$ et $Q \in Prem(BR)$**

b.3) Rappel: Définition d'une démonstration, Conventions

On rappelle ci-dessous les notations, définitions et conventions autour de la notion de démonstration (issu de [GONDRAN89]):

* une démonstration (ou preuve, ou déduction) est une suite finie de phrases p1, ..., ps du système formel dans lequel chaque phrase pi est un axiome ou bien se déduit par une règle de déduction à partir des phrases précédentes pj avec j < i.

* une déduction à partir d'un ensemble fini Hn de phrases h1, ... , hn est une suite finie de phrases p1, ... , ps du système formel dans laquelle chaque phrase pi est un axiome, ou bien une des phrases de Hn, ou bien se déduit par une des règles de déduction à partir des phrases précédentes pj avec j <i.

Exemple: Soit le SF = {Voc, $\mathcal{RF}$, BR $\cup$ EC, $\mathcal{RD}$}.avec EC = $\emptyset$

et BR= {$A \rightarrow B$, $B \wedge C \rightarrow D$, $E \rightarrow F$, $F \wedge A \rightarrow G$, $G \rightarrow \neg D$, $I \wedge G \wedge C \rightarrow H$}

On pose F = {A,C,E}, montrons que F |— D et F |— $\neg$D

$$\dfrac{\dfrac{\dfrac{A^{1} \quad A \rightarrow B^{2}\,(E\rightarrow)}{B^{3}} \quad C^{4}\,(I\wedge)}{B \wedge C^{5} \quad B \wedge C \rightarrow D^{6}\,(E\rightarrow)}}{D^{7}} \qquad \dfrac{\dfrac{\dfrac{E^{8} \quad E \rightarrow F^{9}\,(E\rightarrow)}{F^{10}} \quad A^{11}\,(I\wedge)}{F \wedge A^{12} \quad F \wedge A \rightarrow G^{13}\,(E\rightarrow)}}{\dfrac{G^{14} \quad G \rightarrow \neg D^{15}\,(E\rightarrow)}{\neg D^{16}}}$$

Les numéros en exposant des expressions font référence aux indices des phrases constituant la démonstration.

On distingue dans notre exemple:
- les axiomes: p2, p6, p9, p13, p15
- les hypothèses p1 (p11), p4, p8
- les phrases déduites: p3, p5, p7, p10, p12, p14, p16

Remarques/ Conventions:
* Deux phrases de numéros différents peuvent représenter la même expression, si elle apparait deux fois dans la démonstration.
Exemple: p1 et p11 représentent toutes les deux le littéral A.

* Les dérivations sont toujours de la forme: $p_i \quad p_j$ (avec $j > i$)

$$\frac{}{p_{j+1}}$$

avec p_i une prémisse, p_j une prémisse, une production ou une contrainte et p_{j+1} une prémisse

b.4) Conservation des capacités déductives lors de la transformation

Soit donc une SF du type décrit au a) : SF1 = {Voc, $\mathcal{RF}$, BR ∪ EC, $\mathcal{RD}$}, on lui associe grâce à l'algorithme de réécriture un autre SF, noté SF2, caractérisé par :
SF2 = {Voc ∪ {⊥}, $\mathcal{RF}$, BR ∪ EC', $\mathcal{RD}$ − {(E⇒$_2$)}}.
D'autre part, on pose **BC = BR ∪ EC, BC' = BR ∪ EC'**.

Nous allons démontrer que pour la classe des hypothèses initiales valides, les deux systèmes fournissent "globalement" le même résultat. Pour cela, on montre d'abord que:

Résultat 1: L'ensemble de construction des hypothèses initiales valides avant et après traduction est identique, ie $\mathbf{MNR_{BC} = MNR_{BC'}}$

Démonstration
En effet, rappelons que si $p \in MD_{EC}$, alors il existe dans EC une contrainte liant p et un élément, noté q, de MD_{BR}.
Or, dans tous les cas l'algorithme réécrit au moins dans EC' une contrainte entre p et q, donc $p \in MD_{EC'}$ donc $MD_{EC'} \supset MD_{EC}$.
De plus, il ne crée pas de nouvelle contrainte entre deux mots non reliés auparavant, et ne fait qu'introduire ⊥,
Donc $MD_{EC'} = MD_{EC} \cup \{\perp\}$ d'où $MD_{BC'} = MD_{BC} \cup \{\perp\}$.
De plus, il ne crée pas de nouveau mot autre que ⊥, donc $MO_{BC'} = MO_{BC} \cup \{\perp\}$ donc $MNR_{BC} = MNR_{BC'}$.

On va commencer maintenant à s'intéresser aux capacités déductives des deux systèmes.

Résultat 2: Si h1 est une hypothèse initiale valide et qu'il existe une contrainte logique entre h1 et h2, alors h2 est une hypothèse initiale valide, ce qui s'écrit: **Si Abs(h1) ∈ MNR_{BC} et si EC(⇒, h1, h2)**
Alors Abs(h2) ∈ MNR_{BC}

Démonstration
Supposons que $Abs(h2) \notin MNR_{BC}$,
alors $Abs(h2) \in MD_{BC}$

c'est-à-dire Abs(h2) ∈ MD_{BR} ou Abs(h2) ∈ MD_{EC}

Or, si Abs(h2) ∈ MD_{BR} alors Abs(h1) ∈ MD_{EC}
puisqu'il existe une contrainte logique entre h1 et h2 (définition de MD_{EC}),
ce qui est contradictoire avec Abs(h1) ∈ MNR_{BC}

Sinon, si Abs(h2) ∈ MD_{EC} , c'est qu'il existe une contrainte entre h2 et un nouveau littéral h3 (Abs(h3) ∈ MD_{BR} et $\mathcal{R}$(EC, ⇒, h2, h3)). D'où on sait qu'on a également une contrainte entre h1 et h3 (ie $\mathcal{R}$(EC, ⇒, h1, h3), car $\mathcal{R}$ est transitif, d'où Abs(h1) ∈ MD_{EC} , ce qui est également impossible.

Donc Abs(h1) ∈ MNR_{BC} et EC(⇒, h1, h2) ⇒ Abs(h2) ∈ MNR_{BC}
donc les contraintes sont soit entre littéraux initiaux, soit entre littéraux non initiaux, mais jamais "mixtes".

Résultat 3 : Conservation du "pouvoir déductif" lors de la réécriture:
Soit SF1 un SF du type décrit au a) et SF2 le SF correspondant après réécriture des contraintes logiques,
soit H un ensemble de littéraux tel que $MNR_{BC} \supset$ Abs(H),
soit p ∈ Cons(BR) ∪ Prem(BR) ou Abs(p) ∈ MNR_{BC} [5] alors
H |—SF1 p ⇔ ∃ H' , $MNR_{BC'} \supset$ Abs(H') et H' |—SF2 p

Démonstration

⇒:
Soit s1, s2, ... , sn une démonstration de p à partir de H dans SF1 (on a donc sn = p et sn ∈ Cons(BR) ∪ Prem(BR) ∪ MNR_{BC} ∪ Opposé(MNR_{BC})), montrons par récurrence que: **∀ sj tq Cons(BR) ∪ Prem(BR) ∪ MNR_{BC} ∪ Opposé(MNR_{BC}) ⊃ sj, sj est démontrable dans SF2 à partir de H' , avec $MNR_{BC'} \supset$ Abs(H').**

a) la propriété est vraie pour j = 1
En effet, aucune règle de déduction de SF1 ne permet de combiner les axiomes entre eux. Donc la première étape de la démonstration est forcément de la forme :

s1 s2
------- (X) avec (X) ∈ {(E→), (E⇒1), (E⇒2)}
s3

donc s1 est une hypothèse initiale, littéral, ie un élément de H,
donc s1 appartient à MNR_{BC}.

---

[5] donc p est soit un littéral apparaissant dans les productions issues directement des règles, soit une hypothèse initiale valide.
C'est-à-dire un littéral correspondant à une expression (Entité comparateur Valeur) déductible par les règles, ou pouvant être présent en base de faits initiale.

Posons que H' ⊃ H, on a bien que s1 ∈ H' , donc que s1 est démontrable à partir de H' dans SF2. Notons par ailleurs que H' ⊃ H n'est pas contradictoire avec la condition $MNR_{BC'} \supset Abs(H')$ puisque $MNR_{BC} = MNR_{BC'}$ et $MNR_{BC} \supset Abs(H)$.

b) supposons que les s_i , $i \leq k$ tq $s_i \in$ Cons(BR) ∪ Prem(BR) ∪ MNR_{BC} ∪ Opposé(MNR_{BC}) soient démontrables dans SF2 et que $s_{k+1} \in$ Cons(BR) ∪ Prem(BR) ∪ MNR_{BC} ∪ Opposé(MNR_{BC}) ,
montrons que s_{k+1} est démontrable dans SF2.
Deux cas peuvent se produire:

- soit $s_{k+1} \in H$, auquel cas $s_{k+1} \in H'$, donc est démontrable dans SF2 à partir de H',
- soit s_{k+1} est inféré. Ce qui peut se produire de quatre manières différentes:

* soit

$$\frac{s_i \quad s_k}{s_{k+1}} \ (I\wedge) \text{ avec } i < k$$

Dans ce cas, on a $s_{k+1} = s_i \wedge s_k$
donc $s_{k+1} \neq p$
Or s_{k+1} est une phrase de la démonstration de p,
donc ∃r' tq Prem(r') ⊃ s_{k+1} (on est en train de démontrer que la partie prémisse d'une production nécessaire à la déduction de p est vraie)
donc $s_i \in$ Prem(BR) et $s_k \in$ Prem(BR)
donc, par hypothèse de récurrence, s_i et s_k sont démontrables dans SF2
donc, puisque (I∧) appartient aux règles de déduction de SF2, s_{k+1} est démontrable dans SF2.

* soit

$$\frac{s_i \quad s_k}{s_{k+1}} \ (E\rightarrow) \text{ avec } i < k \text{ et } s_k \text{ une production de BR notée R1}$$

Dans ce cas, s_i = Prem(R1), c'est donc une conjonction d'éléments de Prem(BR). Elle est de la forme $s_i = s_{i0} \wedge s_{i1} \wedge \ldots \wedge s_{ip}$ avec i0 < i, i1 < i, ..., ip < i
et $(s_{i0}, s_{i1}, \ldots, s_{ip}) \in (Prem(BR))^{ip}$
Tous les s_{ij} sont démontrables dans SF2 à partir de H', donc s_i est démontrable dans SF2 (à partir de H').
De plus R1 appartient à SF2, donc l'inférence est également possible dans SF2, donc s_{k+1} est démontrable dans SF2 à partir de H'.

* soit s_i s_k

-------- (E⇒j) avec i < k, j = 1 ou 2, et s_k un élément de EC

s_{k+1}

Dans ce cas :

- soit $Abs(s_{k+1}) \in MNR_{BC}$, auquel cas on inclut s_{k+1} dans H',
- soit $Abs(s_{k+1}) \notin MNR_{BC}$.

On sait qu'on a $EC(\Rightarrow, s_i, s_{k+1})$

[*] Or $EC(\Rightarrow, s_i, s_{k+1})$

$\Rightarrow Abs(s_i) \notin MNR_{BC}$ (d'après Résultat 2) $\Rightarrow s_i$ est déduit. Donc:

- soit s_i est déduit par (E→),

$\Rightarrow s_i = Cons(r)$ avec $r \in BR$,

$\Rightarrow s_i \in Cons(BR)$,

Et on a $EC(\Rightarrow, s_i, s_{k+1})$

$\Rightarrow s_{k+1} \notin Cons(BR)$ (règles l2 sur les contraintes logiques).

Or on sait que $Abs(s_{k+1}) \notin MNR_{BC}$ et $s_{k+1} \in Cons(BR) \cup Prem(BR) \cup MNR_{BC} \cup Opposé(MNR_{BC})$ $\Rightarrow s_{k+1} \in Prem(BR)$.

Or si $EC(\Rightarrow, s_i, s_{k+1})$, $s_i \in Cons(BR)$ et $s_{k+1} \in Prem(BR)$,
alors la contrainte s'écrit $s_i \Rightarrow s_{k+1}$ dans EC' (cf la remarque du paragraphe b.1),

De plus, s_i est démontrable dans SF2 puisque i < k et $s_i \in Cons(BR)$
donc s_{k+1} est démontrable dans SF2 (à partir de H').

- soit s_i est déduit par (E⇒j) avec j = 1 ou 2, ie qu'on a

s_{k0} s_{i-1}

---------- (E⇒j) avec k0 < i-1, j = 1 ou 2, et s_{i-1} un élément de EC

s_i

donc on a $EC(\Rightarrow, s_{k0}, s_i)$

$\Rightarrow EC(\Rightarrow, s_{k0}, s_{k+1})$ (transitivité de "⇒" dans EC) et s_{k0} démontrable dans SF1.

On peut reprendre avec s_{k0} le même raisonnement qu'avec s_i à partir de [*]
Dans tous les cas, s_{k+1} est démontrable sauf si s_{k0} est lui-même déduit grâce à un autre littéral désigné par s_{k1} grâce à une relation du type
$EC(\Rightarrow, s_{k1}, s_{k0})$ avec k1 < k0.
On a alors encore aussi $EC(\Rightarrow, s_{k1}, s_{k+1})$,
et on peut continuer avec s_{k1} en repartant de [*] ...

Or le nombre de littéraux est fini
donc la seule possibilité pour que s_{k+1} ne soit pas démontrable est d'avoir un circuit dans la suite des déductions. On aurait alors:
s_{k+1} non démontrable car

$\exists i < k+1$ tel que $EC(\Rightarrow, s_i, s_{k+1})$ et

$\exists k_0 < i$ tel que $EC(\Rightarrow, s_{k0}, s_i)$ et

... et

$\exists k_p < k_{p-1}$ tel que $EC(\Rightarrow, s_{kp}, s_{kp-1})$ et

$\exists m < k_p$ tel que $s_m = s_{k+1}$ et $EC(\Rightarrow, s_m, s_{kp})$

Ce qui revient à dire qu'on ferait une série de déduction pour arriver à démontrer un littéral déjà démontré précédemment, ce qui est absurde. (De plus, on sait par hypothèse de récurrence que s_m est démontrable dans SF2, donc $s_{k+1} = s_m$ est aussi démontrable dans SF2).

Donc
- s_{k+1} est démontrable dans SF2 à partir de H' avec
$H' = H \cup \{s_i, i < k, tq\ s_i \notin H$ et $Abs(s_i) \in MNR_{BC}\}$.
- On a bien **$MNR_{BC'} \supset Abs(H')$** (puisque $MNR_{BC} = MNR_{BC'}$ d'après Résultat 1).

$\Leftarrow$:
La réciproque est évidente: SF2 étant une version "dégradée" de SF1, tout littéral démontrable dans SF2 à partir d'un ensemble d'hypothèses l'est également dans SF1 à partir des mêmes hypothèses.

b.5) Conséquence sur l'ensemble des Mots Non Récepteurs

Nous allons voir qu'on peut faire, dans le nouveau système, *à peu de choses près*, l'assimilation entre une **hypothèse initiale** d'une démonstration (ie un élément h tq $Abs(h) \in MNR_{BC}$) et un **élément n'appartenant pas à la partie conséquent** d'une production (issue directement d'une règle ou de la réécriture d'une contrainte logique).

Plus précisément, soit SF1 = $\{Voc, \mathcal{RF}, BR \cup EC, \mathcal{RD}\}$ un SF du type décrit au a) et SF2 = $\{Voc \cup \{\bot\}, \mathcal{RF}, BR \cup EC', \mathcal{RD} - \{(E \Rightarrow_2)\}\}$ le SF associé par l'algorithme.
On rappelle qu'on note $BC = BR \cup EC$ et $BC' = BR \cup EC'$.

On étend naturellement à EC' et BC' l'opérateur Cons. On a:
Résultat 4: **Si $p \in MNR_{BC}$[6] Alors $p \notin Cons(BC')$ et $\neg p \notin Cons(BC')$**

[6] ou $p \in MNR_{BC'}$, puisqu'il y a identité entre les deux ensembles (cf Résultat 1)

Démonstration
En effet,
d'une part on sait que $p \notin MD_{BR}$
donc $p \notin Cons(BR)$ et $\neg p \notin Cons(BR)$.
d'autre part, si il existe $\mathcal{R}(EC, \Rightarrow, p, q)$, alors on sait que $q \notin MD_{BC}$ (sinon $p \in MD_{EC}$)
donc $q \notin Cons(BR)$ et $\neg q \notin Cons(BR)$.
donc si s1 désigne la partie droite d'une des contraintes en question, et s2 la partie gauche, on a:
s1 = p ou s1= $\neg$p ou s1 = q ou s1 = $\neg$q, et de même pour s2
donc s1 $\notin$ Cons(BR), et s2 $\notin$ Cons(BR)
donc, d'après l'algorithme, la contrainte s1 $\Rightarrow$ s2 s'écrit s1 $\wedge \neg$s2 $\Rightarrow \perp$ (règle 3)
donc ni s1 ni s2 n'apparaissent en partie conséquent.
donc ni p ni $\neg$p n'apparaissent en partie conséquent d'un élément de EC'
donc $p \notin Cons(BC')$ et $\neg p \notin Cons(BC')$.

Toutefois, la réciproque n'est pas vraie, comme nous le verrons sur l'exemple de la partie suivante.

b.6) Conclusion

Je voudrais ici résumer les intérêts de la manipulation présentée, et justifier la complexité des définitions de MNR_{BC} et MD_{BC}, étant donnée notre contexte de travail.

Les SF que nous manipulons sont issus de la logique Zéro Plus. Néanmoins, par rapport à ce cadre, nos systèmes présentent deux différences notables:

- La première, on l'a déjà dit, est qu'ils contenaient à la fois des productions et des implications logiques. Pour éviter d'avoir à gérer, dans la phase de vérification, deux catégories d'objets différents, nous avons transformé les secondes en premières, sans pour autant changer les capacités déductives du système. Les règles de déduction étant les mêmes sur les deux ensembles BR et EC', on peut maintenant les réunir dans un même formalisme. **On pourra donc confondre dans EC' la contrainte $p \Rightarrow q$ avec la production $p \rightarrow q$.**

- La seconde est la suivante. On sépare généralement, en logique Zéro Plus, les entités "initiales" (celles pouvant figurer en base de faits initiale) du reste de l'ensemble des entités. Elles correspondent presque toujours (sauf dans [HERY85] et [BEAUVIE89]) aux entités non déductibles, ie n'apparaissant pas en conséquent de règle (cf [CHARLES90], [PIPARD87], [ROUSSET88a],...). Or, dans notre cas:

a) Les (littéraux représentant des triplets contenant des) entités initiales apparaissaient en partie conséquent d'implications

exemple: Dans un système de gestion de cuve en fonction de l'état de pompes en entrée, dites pompes primaires, les contraintes exprimant la monovaluation de ces entités sont du type:

POMPE_PRIMAIRE_00i = 'ouvert' ⇒ POMPE_PRIMAIRE_00i <> 'fermé'

ce qui rendait invalides les définitions classiques.

D'où la réécriture de ces contraintes sous la forme:

POMPE_PRIMAIRE_00i = 'ouvert' ∧ POMPE_PRIMAIRE_00i= 'fermé' ⇒ ⊥

ce qui amène au résultat 4 rétablissant la propriété qu'une entité initiale est non déductible.

b) nous avons des littéraux clairement non initiaux et n'apparaissant pourtant qu'en partie prémisse de production (b1) , voire même qu'en partie prémisse dans les productions et les contraintes logiques (b2).

exemple: admettons que le système gère la pression d'une cuve. On a donc une série de règles concluant sur PRESSION_CUVE = 5, 10 ou 20.

b1 - Admettons alors que l'on ait aussi dans les règles l'expression PRESSION_CUVE > 10, qui n'apparait qu'en partie prémisse.

Auquel cas le mot correspondant à ce triplet (appelons-le m1) n'apparait jamais en conséquent de production. Il ne fait pourtant pas partie des hypothèses initiales valides.

Toutefois, dans ce cas, la traduction nous assure que l'on a dans EC la contrainte

PRESSION_CUVE = 20 ⇒ PRESSION_CUVE > 10,

donc que m1 est "déductible grâce aux contraintes".

La contrainte sera donc réécrite dans EC' tel quel, ou retournée si elle se présentait sous la forme: ¬(PRESSION_CUVE > 10) ⇒ PRESSION_CUVE <> 20,

On explicite la "déductibilité" de m1 qui devient dans EC' un littéral déductible comme les autres. **On a donc ainsi à nouveau rétabli dans EC' l'égalité:**

littéral non initial = littéral déductible

b2 - **Mais** imaginons maintenant que l'on ait aussi dans ce système l'expression PRESSION_CUVE > 100 qui n'apparait aussi qu'en prémisse de règle sans que PRESSION_CUVE ≤ 100 ne soit présent dans le système.

Etant donné les valeurs déductibles pour l'entité PRESSION_CUVE (c'est-à-dire 5, 10 et 20), on n'a pas dans EC de contrainte qui conclue sur PRESSION_CUVE > 100, mais seulement des relations du type:

PRESSION_CUVE = 5, 10, 20 ⇒ PRESSION_CUVE ≤ 100.

Fidèle au principe de ne réécrire que des implications qui puissent réellement servir dans une démonstration, nous avons réécrit la contrainte ci-dessus sous sa seule forme exploitable, c'est-à-dire:

PRESSION_CUVE = 5, 10, 20 ∧ PRESSION_CUVE > 100 ⇒ ⊥

Du coup, le littéral représentant le triplet PRESSION_CUVE > 100 s'avère ne jamais apparaître en conséquent dans tout le système, tout en étant clairement une hypothèse initiale non valide. Il est non déductible mais non initial.

C'est ce type de situation, à laquelle la transformation ne change rien, qui amène la relative complexité des définitions de MNR_{BC} et MD_{BC}.

C'est elle qui conduit à n'avoir, après transformation, que l'inclusion de MNR_{BC} dans (et non l'égalité de MNR_{BC} avec) l'ensemble des mots non déductibles.

c) Le nouveau système formel après transformation

A l'issue de la réécriture, et en confondant productions et contraintes (ce qui licite, comme nous venons de le démontrer), on aura un système formel du type:

1) Vocabulaire = $\{ \neg, \wedge, \rightarrow, MO_{BC} \} \cup \{\bot\}$

2) Règles de formation des phrases: $\mathcal{RF}$

- si m est un mot courant (c'est-à-dire un élément de MO_{BC}), m et $\neg$m sont des phrases appelées littéraux.
- si p et q sont des littéraux ou des prémisses, p, q et $p \wedge q$ est une prémisse
- si p est une prémisse et q un littéral ou $\bot$, $p \rightarrow q$ est une production

3) Les Axiomes sont constitués d'un ensemble BC de productions, (encore appelées par extension des règles, ou des règles-axiomes)

4) Les règles de déduction sont : $\mathcal{RD}$

$(I\wedge)$: $\dfrac{A \quad B}{A \wedge B}$ $\qquad$ $(E\rightarrow)$: $\dfrac{A \qquad A \rightarrow B}{B}$

On désigne par MD_{BC} l'ensemble des mots déductibles du système après réécriture. On sait que cet ensemble ne varie pas au cours de la réécriture. Toutefois, pour le spécifier à l'aide du système réécrit, on doit écrire:

MD_{BC} = Abs(Cons(BC)) $\cup$

$\{Abs(p); p \notin Cons(BC)$ et $p \wedge q \rightarrow \bot \in BC$ ou $p \wedge \neg q \rightarrow \bot \in BC$ et $q \in MD_{BC}\}$

Le deuxième ensemble ci-dessus est l'ensemble des mots non déductibles et non récepteurs de BC: ce sont des mots "parasites" puisqu'ils ne peuvent ni être déduits, ni faire partie d'un ensemble d'hypothèses initiales valides: on note cet ensemble MP_{BC} (Mots Parasites de BC).

De même, on désigne par MNR_{BC} l'ensemble des mots non récepteurs du système après réécriture. On sait que $MNR_{BC} = MO_{BC} - MD_{BC}$, mais on peut aussi écrire: $MNR_{BC} = Abs(Prem(BC)) - MP_{BC}$

Rappelons que MNR_{BC} est l'ensemble des mots à partir desquels on construit les hypothèses initiales valides des démonstrations dans SF_{BC}.

4) Objectif

Terminons enfin nos préliminaires par la présentation, dans deux formalismes, de la propriété que nous chercherons à vérifier dans le chapitre suivant.
Cette formalisation nécessite d'abord un rappel sur la logique trivaluée de Herbrand.

a) Présentation de la Logique Trivaluée de Herbrand

a.1) Introduction, interprétations de Herbrand

La sémantique des systèmes à base de connaissances repose, contrairement à la Logique Classique, sur la possibilité pour une entité d'être vraie, fausse ou inconnue (indéterminée). Aussi un système de vérification des SBC doit-il tenir compte de ces trois valeurs possibles, et mettre en oeuvre des méthodes fondées sur la logique correspondante, la Logique Trivaluée, que nous allons sommairement présenter ici.

Les définitions que nous allons exposer sont principalement extraites de [DELAHA87]. Elles sont conçues pour des SF construits sur un langage d'ordre un, mais s'appliquent aussi à un SF de type propositionnel[7].
Soit donc L un langage, on note de manière classique :

- Ter(L): l'ensemble des termes de L : ie les constantes, les variables et toute combinaison valide de symboles fonctionnels, de constantes et de variables
- Ato(L): l'ensemble des formules atomiques ou atomes de L: ie l'ensemble des liens entre termes par l'intermédiaire d'une relation de la signature
- Lit(L): l'ensemble des littéraux de L: ie Ato(L) $\cup \neg$Ato(L)
- For(L): l'ensemble des formules de L : ie l'ensemble des littéraux et des expressions du type $A \wedge B$, $A \rightarrow B$, $A \Rightarrow B$, $\neg A$, $\forall A$, etc ... , avec A et B des formules.

On appelle L-base de Herbrand et on note her(L) l'ensemble des atomes sans variable de L. On appelle L-base de Herbrand double et on note Her(L) l'ensemble de tous les littéraux sans variables de L:
$Her(L) = her(L) \cup \neg her(L)$

On appelle L-interprétation de Herbrand trivaluée (encore appelée interprétation partielle) la donnée d'une partie i de Her(L) ne contenant jamais un atome et sa négation. On note Iht(L) l'ensemble des L-interprétations de Herbrand trivaluées.

Les éléments maximaux (pour l'inclusion) de Iht(L) sont les interprétations pour lesquelles: $\forall$ At $\in$ her(L), At $\in$ i ou $\neg$At $\in$ i
sont appelés L-interprétation bivaluées. Leur ensemble est noté Ihb(L).

---

[7] Rappelons pour mémoire que l'ensemble des propositions, encore appelées formes propositionnelles (cf [LALEUF90], p 5) sur un ensemble A est défini comme l'ensemble $T_{\Sigma}[A]$ des termes construits à partir de la signature
$\Sigma = \{\wedge, \vee, \neg, \Rightarrow\}$, les éléments de A étant appelées variables propositionnelles ou formules premières (cf [LAL&SA86], p 18 ou chapitre 1 [LALEME90]).

a.2) Valeurs de vérité relativement à une interprétation

Soit i ∈ Iht(L), on définit la fonction valeur de vérité vv_i qui à toute formule sans variable libre f associe $vv_i(f) \in \{V, F, I\}$:

a) si f est atomique

$vv_i(f) = V$ si $f \in i$
$vv_i(f) = F$ si $\neg f \in i$
$vv_i(f) = I$ sinon

b) $vv_i(F1 * F2) = * (vv_i(F1), vv_i(F2))$ pour $* \in \{\wedge, \Rightarrow^8, \rightarrow\}$
où $\wedge(.,.)$,$\Rightarrow(.,.)$, $\rightarrow(.,.)$ sont des applications de $\{V, F, I\}^2$ dans {V, F, I} définis par les tables suivantes:

$\wedge$	V	F	I
V	V	F	I
F	F	F	F
I	I	F	I

$\Rightarrow$ [9]	V	F	I
V	V	F	I
F	V	V	V
I	V	I	I

$\rightarrow$	V	F	I
V	V	F	F
F	V	V	V
I	V	V	V

c) $vvi(\neg f) = \neg (vvi(f))$ où
$\neg$ (.) est l'application de {V, F, I} dans {V, F, I} définie par:

$\neg (V) = F$, $\neg(F) = V$, $\neg (I) = I$

[8] Il s'agit ici du "⇒" complet, ie celui exploité par les deux règles (E⇒1) et (E⇒2). Il serait donc incorrect de prendre la table pour construire l'interprétation des formules contenant "⇒" des SF du type SF2 décrit en 3)b.4 (voir le Nota Bene du 3)b.2)).

[9] idem note ci-dessus

On donne également, bien que cela ne concerne pas directement les SF qui nous intéressent, la table de vérité du "$\vee$":

$\vee$	V	F	I
V	V	V	V
F	V	F	I
I	V	I	I

a.3) Commentaires

- Dans notre cas, l'ensemble des atomes est égal à l'ensemble des termes (dans le formalisme d'ordre un, les symboles $\wedge$, $\vee$, $\neg$, $\Rightarrow$, et $\rightarrow$ ne font pas partie de la signature du langage. Notre signature ne possède donc pas de symbole relationnel). Donc l'ensemble des atomes sans variable correspond à l'ensemble des termes sans variable, ie à l'ensemble des mots courants (MO_{BC})

Une interprétation partielle i est donc une partie de $MO_{BC} \cup$ Opposé(MO_{BC}) qui n'est pas contradictoire, ie tel que i $\cap$ Opposé(i) = $\varnothing$.
On peut la confondre avec la restriction de vv_i aux atomes. Dans ce cas, **i devient une fonction de MO_{BC} dans {V, F, I}.**

- L'algorithme de réécriture du b) a introduit "$\perp$" et éliminé "$\Rightarrow$". Plus précisément, on sait donc que l'on peut trouver dans BC' des formules du type $A \rightarrow \perp$.
La sémantique attachée à cette formule étant "A ne peut être vrai", on étend toute fonction vv_i à $\perp$ de la manière suivante: $\mathbf{vv_i(\perp) = F}$[10].
De cette manière, on a $vv_i(A \rightarrow \perp) = V$ si $vv_i(A) = F$ ou $vv_i(A) = I$.

- Concernant le "non" ($\neg$), remarquons que la sémantique choisie:
+ respecte la loi de réécriture (E$\neg\neg$) : $\neg\neg A = A$
En effet $vv_i(\neg\neg A) = \neg(vv_i(\neg A)) = \neg(\neg(vv_i(A)))$.
Or: $\neg(\neg(V)) = \neg(F) = V$, $\neg(\neg(F)) = \neg(V) = F$, $\neg(\neg(I)) = \neg(I) = I$
Donc $\forall x \in \{V, F, I\}$, $\neg(\neg(x)) = x$
Donc $vv_i(\neg\neg A) = vv_i(A)$

+ tout en ne garantissant pas la tautologie $A \vee \neg A$.
En effet, si $A \notin i$, $vv_i(A) = vv_i(\neg A) = I$ donc $vv_i(A \vee \neg A) = I$
Exemple: Le mot m1 représentant l'expression (EI1 = 5) peut valoir:
- Vrai : l'entité EI1 a la valeur 5 dans le système

---

10 a) voir la note concernant $\perp$ lors de son introduction.
b) notons que c'est la fonction valeur de vérité vv_i que l'on étend de façon systématique à $\perp$, ce qui revient à considérer que "$\perp$" ne fait pas partie de MO_{BC}, donc ne peut faire partie de i. Nous adoptons cette convention car elle simplifie les concepts en isolant le cas particulier de "$\perp$". D'autre part, sémantiquement, MO_{BC} contient les mots "concrets" apparaissant dans la base de connaissances, pas les "facilités d'écriture".

- Faux : l'entité EI1 a une valeur différente de 5 dans le système (la valeur 2 par exemple)

- Indéterminé: L'entité EI1 n'a pas de valeur dans le système, elle est inconnue pour le cas considéré

Remarque: Avec cette sémantique, $\neg m1$ représente l'expression (EI1 $\neq$ 5), la sémantique du $\neq$ étant: "Possède une valeur connue, et cette valeur est différente de", comme c'est le cas dans la plupart des moteurs d'inférence.

De même si m représente l'expression (E > k), $\neg m$ représente (E $\leq$ k) et ainsi de suite pour les autres comparateurs logiques.

- On peut enfin remarquer que si $A \Rightarrow B$ correspond en fait à $\neg A \vee B$, $A \rightarrow B$ **ne peut se réécrire en fonction des autres opérateurs** (cf [DELAHA87], p 23).

Nous allons maintenant arriver à la définition de la propriété qui nous intéresse, que nous pourrons aborder tant sous l'angle syntaxique que sémantique.

b) Cohérence et BC-Cohérence

b.1) Définition

Un SF est cohérent s'il n'est pas possible de démontrer un théorème et son contraire, ie p et $\neg p$.

Cette définition implique de fait également l'impossibilité de dériver l'atome de contradiction $\perp$, puisque pour dériver $\perp$, il faut avoir à la fois A et $A \rightarrow \perp$ (et appliquer le modus ponens), ie sémantiquement avoir "A est vrai" et "A n'est pas vrai".

D'autre part, nous avons vu:

- au chapitre 1, 1)d), qu'une base de règles était cohérente si aucun processus d'inférence à partir d'une base de faits initiale valide ne permettait la déduction de faits contradictoires,
- au chapitre 2, 1), qu'une base de faits initiale valide était composée de couples (E, val) avec E une entité désignée initiale par l'expert et n'apparaissant jamais en partie droite d'une règle de BR (entité non déductible), et val une valeur".
- au chapitre 2, 3)a.2), qu'aux différentes formes d'apparition des entités non déductibles dans la base de règles correspondaient dans le SF les littéraux non récepteurs.

Donc pour vérifier la cohérence d'une base de règles, nous allons nous intéresser à la cohérence, non pas du SF issu de la traduction, mais de **tous les SF** constitués à partir: . du SF issu de la traduction, c'est-à-dire d'un SF du type décrit au 3)c)

. d'un ensemble non intrinsèquement contradictoire de **littéraux** se rapportant à une entité initiale, sachant que ces littéraux sont alors forcément **non récepteurs** (ie, *a peu de choses près,* définis à partir de mots n'apparaissant pas en partie droite des productions et des contraintes réécrites).

On parlera alors de **BC-Cohérence du SF concerné.**

Plus précisément, on dira que SF_{BC} est <u>BC-cohérent</u> si:
$\forall BF$ tq $MNR_{BC} \supset$ Abs(BF) et $BF \cap$ Opposé(BF) = $\varnothing$,
$\forall p$, non($BF \vdash_{BC} p \wedge \neg p$) et non($BF \vdash_{BC} \perp$)

"Tout système formel constitué de SF_{BC} et d'un ensemble cohérent d'axiomes (d'hypothèses) supplémentaires appartenant à $MNR_{BC} \cup$ Opposé(MNR_{BC}) doit être cohérent".
Autrement dit, à partir d'hypothèses initiales valides, on ne peut déduire $p \wedge \neg p$ ou $\perp$ à partir des axiomes du SF.

On définit aussi la cohérence par rapport à un ensemble de mots:
On dit que SF_{BC} est **<u>cohérent par rapport à un ensemble de mots E</u>** si:
$\forall BF$ tq E $\supset$ Abs(BF) et $BF \cap$ Opposé(BF) = $\varnothing$,
non ($BF \vdash_{BC} p \wedge \neg p$) et non ($BF \vdash_{BC} \perp$)

Autrement dit, ici, les hypothèses initiales valides pour les démonstrations dans BC doivent être des littéraux construits à partir de mots de E.
On notera **$Valid_E$(BF)** la propriété: **"E $\supset$ Abs(BF) et $BF \cap$ Opposé(BF) = $\varnothing$".**
La propriété devient: **$\forall BF$ tq $Valid_E$(BF), non ($BF \vdash_{BC} p \wedge \neg p$) et non ($BF \vdash_{BC} \perp$)**
Donc SF_{BC} est BC-cohérent s'il est cohérent par rapport à MNR_{BC}.

<u>REMARQUES IMPORTANTES</u>:
1°) De façon **systématique,** l'ensemble **E** considéré sera **d'intersection vide** avec l'**ensemble des mots déductibles** du SF considéré, ie: $E \cap MD_{BC} = \varnothing$. (Par conséquent, si $Valid_E$(BF), alors $BF \cap MD_{BC} = \varnothing$).
On vérifiera le maintien de la validité de cette hypothèse lors des transformations apportées au système initial.

2°) **En regard de la BC-Cohérence, les deux systèmes avant et après réécriture, sont équivalents.**
En effet, soit SF1 = {Voc, $\mathcal{RF}$, BR $\cup$ EC, $\mathcal{RD}$} un SF du type décrit au a)
et SF2 = {Voc $\cup$ {$\perp$}, $\mathcal{RF}$, BR $\cup$ EC', $\mathcal{RD}$ - {(E$\Rightarrow_2$)}} le SF associé par l'algorithme, on sait que : - $MNR_{BR \cup EC} = MNR_{BR \cup EC'}$ (résultat 1) et que
- si H est un ensemble de littéraux tel que $MNR_{BC} \supset$ Abs(H), alors
si $p \in$ Cons(BR) $\cup$ Prem(BR) $\cup$ MNR_{BC}) alors
$H \vdash_{SF1} p \Leftrightarrow \exists H'$ tq $MNR_{BC'} \supset$ Abs(H') et $H' \vdash_{SF2} p$ (résultat 3).

Donc ($\forall BF$ tq $MNR_{BR \cup EC} \supset$ Abs(BF) et $BF \cap$ Opposé(BF) = $\varnothing$,
$\forall p$, non($BF \vdash_{BR \cup EC} p \wedge \neg p$) et non($BF \vdash_{BR \cup EC} \perp$))
$\Leftrightarrow$($\forall BF$ tq $MNR_{BR \cup EC'} \supset$ Abs(BF) et $BF \cap$ Opposé(BF) = $\varnothing$,

$\forall p$, non(BF $\vdash_{BR \cup EC'} p \wedge \neg p$) et non(BF $\vdash_{BR \cup EC'} \perp$)).
On s'intéressera donc à partir de maintenant à trouver des méthodes de vérification de la BC-Cohérence sur les systèmes réécrits.

b.2) Approche sémantique de la BC-Cohérence

Il est bien connu (Gödel, théorème d'existence des modèles, 1930) que la cohérence d'une théorie est équivalente à l'existence d'un modèle pour cette théorie. Dans le cas de la logique zéro, les théories sont réalisées dans une algèbre de Boole, en se basant sur une interprétation, c'est-à-dire une application de l'ensemble des variables propositionnelles dans {V, F}.

Dans notre cas:

- d'une part, on a vu qu'il faut réaliser notre théorie dans {V, F, I}.

Plus précisément, on cherche une interprétation partielle i qui soit un **modèle fort** de la théorie, ie tel que :
pour toute formule sans variable f de la théorie, on a $vv_i(f) = V$[11]

- d'autre part, on sait qu'on cherche à vérifier la cohérence de tous les SF constitués du SF initial et d'un ensemble non-contradictoire quelconque BF d'éléments de MNR_{BC}. Cela revient donc toujours:

* d'abord à satisfaire BF, ce qui est toujours possible puisque BF est non-contradictoire. Soit $i0_{BF}$ l'application de Abs(BF) dans {V, F} définie par
$i0_{BF}$: Abs(BF) --> {V, F}[12] telle que
$i0_{BF}(x) = V$ si $x \in$ BF et $i0_{BF}(x) = F$ si $\neg x \in$ BF
* puis à construire l'application complémentaire $i1_{BF}$ défini par:
$i1_{BF}$: MO_{BC} - Abs(BF) --> {V, F, I} telle que
si on note i la réunion[13] de $i0_{BF}$ et $i1_{BF}$, on a : $\forall r \in$ BC, $vv_i(r) = V$

De manière évidente, la cohérence d'un SF_{BC} par rapport à un ensemble E se définit de la même façon, la seule différence concernant la restriction sur BF.

---

[11] Par opposition à un modèle faible, défini par $vv_i(f) \neq F$. J.P. Delahaye a montré dans [DELAHA87] tout l'intérêt des modèles forts.

[12] On cherche un modèle fort, donc chaque élément de BF, en tant qu'axiome, doit être réalisé de telle sorte que sa valeur de vérité vale vraie, ce qui implique que $i0_{BF}$ réalise BF dans {V, F} et non dans {V, F, I}.

[13] Plus précisément, i est défini par:
i: MO_{BC} --> {V, F, I} et
si $x \in$ BF, $i(x) = i0_{BF}(x)$
si $x \in MO_{BC}$ - BF , $i(x) = i1_{BF}(x)$

CHAPITRE III: L'algorithme de Vérification de la BC-Cohérence de MELOMIDIA

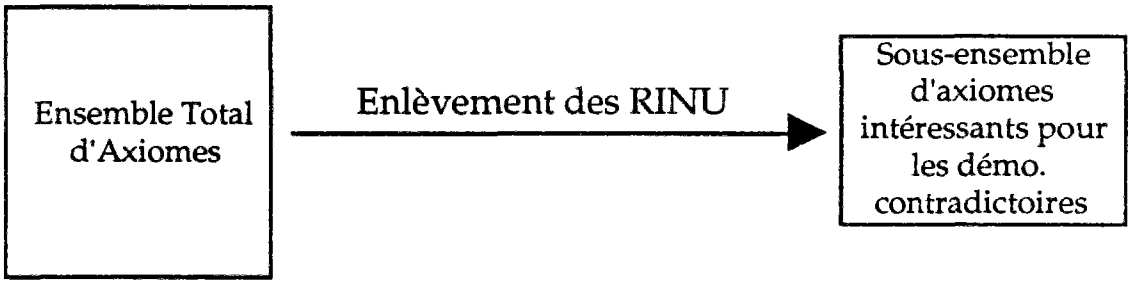
1) Présentation

a) Introduction

Un système formel du type décrit au chapitre précédent, c'est-à-dire issu de la traduction d'une base de règles, possède en général un grand nombre d'axiomes. Ceux-ci contiennent la connaissance (codée) de l'expert dans le domaine considéré. Cette connaissance comporte le plus souvent plusieurs aspects indépendants, une même cause ayant fréquemment plusieurs effet indépendants entre eux.

Une vérification de la BC-Cohérence d'un système formel de ce type doit profiter de cette réalité: plusieurs champs de connaissances sont à étudier, certains potentiellement contradictoires (ceux qui correspondent à des parties de la BR de départ dans lesquelles une entité prend des valeurs différentes suivant le contexte), d'autres clairement non-ambigus (on propage l'effet - toujours le même - d'un phénomène sur un certain nombre de paramètres).

Le premier travail d'un vérificateur de BC-Cohérence est donc de réduire l'ensemble d'axiomes à sa sous-partie minimale potentiellement contradictoire. C'est à dire d'isoler simplement les parties "délicates" de la formalisation de celles totalement non ambigües. Cette opération est appelée l'enlèvement des règles-axiomes inutiles[1]:



Par ailleurs, un système BC-incohérent est un système capable de déduire p et ¬p ou ⊥ à partir d'hypothèses initiales valides[2]. Nous allons donc mettre en place, sur le système d'axiomes réduit, **un mécanisme de remontée, en "chainage arrière", des contradictions potentielles vers les hypothèses initiales.** En remplaçant successivement, dans l'ensemble des règles-axiome, tous les littéraux déductibles par leurs contextes de déduction, c'est-à-dire en supprimant progressivement toutes les étapes intermédiaires entre les hypothèses de départ et la déduction de littéraux contradictoires, nous saurons si les incohérences ont réellement une chance de se produire. En effet, une fois la remontée totalement effectuée, nous savons, s'il reste encore dès axiomes dans le système, quelles combinaisons d'hypothèses valides conduisent à des déductions contradictoires.

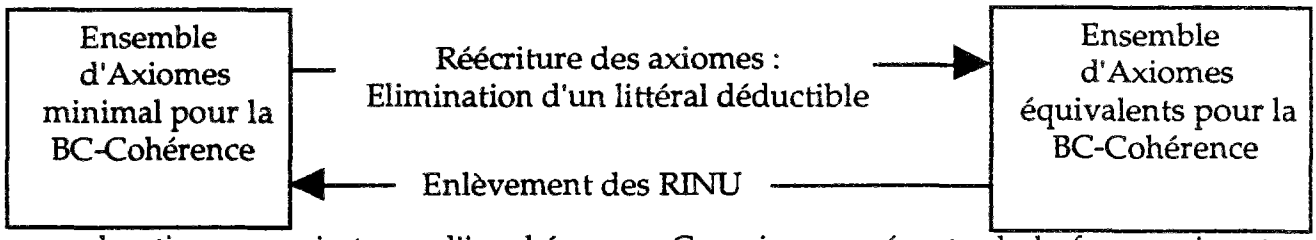
---

[1] Parfois, dans le cas où ce sous-ensemble est vide, ce traitement fournit inmédiatement une réponse positive à notre vérification de la cohérence. C'est le cas pour la base EXTRA41 (990 règles) de l'EDF.

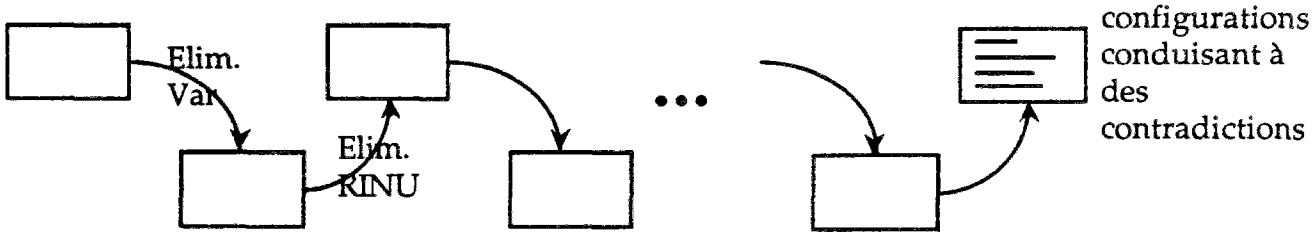
[2] qui sont, nous le rappelons, des littéraux construits à partir de mots non récepteurs, ie de mots non déductibles (voir chapitre précédent).

Cette remontée pourrait conduire le système à envisager des cas inutiles ou impossibles. C'est pourquoi on adjoint au mécanime d'élimination des entités non initiales le mécanisme déjà évoqué de maintien de la minimilaté du nombre d'axiomes présents dans le système.

On a donc un enchaînement de taches du type:



pour aboutir aux conjectures d'incohérences. Ce qui se représente de la façon suivante:



b) Principe de l'élimination des littéraux déductibles

On a déjà dit qu'un SF BC-incohérent est un système capable de déduire ¬p après avoir déduit p, ou de déduire "⊥", à partir d'hypothèses initiales valides. Or, ici:

1°) Nos règles de déduction ne permettent que:
* de déduire le conséquent d'une règle-axiome si les prémisses de cette règle sont vérifiées (E→)
* de rassembler des littéraux existant (déjà déduits ou hypothèses initiales) en vue de satisfaire la partie prémisse d'une règle (I∧),
Donc une démonstration se limite à constituer une partie de prémisse de règle à en déduire le conséquent, puis à utiliser ce conséquent pour constituer la partie prémisse d'une autre règle, en tirer le conséquent, etc ...

2°) On limite les hypothèses valides aux littéraux construits sur des mots non déductibles. Donc un axiome ne pourra intervenir dans une démonstration que si et seulement si chaque littéral composant sa partie prémisse:
- soit appartient à l'ensemble des mots non récepteurs (ie non déductibles)
- soit est déduit par (application de (E→) sur) une autre règle concluant sur ce littéral.

Donc, si une prémisse d'une règle est constituée d'un littéral déductible, on ne change nullement ce que l'on peut démontrer dans le SF en remplaçant cette prémisse par la partie prémisse de la règle déduisant ce littéral. Si plusieurs règles permettent de le déduire, elles constituent autant de possibilités de satisfaire la prémisse de la règle initiale, il faut donc écrire autant de nouvelles règles.

> Donc, pour éliminer toutes les occurences d'un littéral (déductible) V en partie prémisse de règle sans changer les capacités déductives du SF,
il suffit de remplacer chaque occurence en partie prémisse de V (resp. ¬V) par l'union des parties prémisses des règles déduisant V (resp. ¬V).

Si on a r1: A1 → B, r2: A2 → B, r3: A3 → B, r4: C ∧ B → D
On peut remplacer r4 par: r4': C ∧ (A1 ∨ A2 ∨ A3) → D,
c'est-à-dire, avec notre formalisme, par les trois règles:
r4.1: C ∧ A1 → D
r4.2: C ∧ A2 → D
r4.3: C ∧ A3 → D

A l'issue de cette transformation, les littéraux V et ¬V ne sont plus plus présents qu'en conséquent de règle[3].

D'autre part,
1°) le SF tranformé après fermeture transitive n'a plus besoin des axiomes concluant sur le littéral concerné, si ce n'est pour déduire ce littéral ou son opposé.
2°) en terme de BC-Cohérence, chaque couple constitué d'une règle concluant l'une sur le littéral et d'une autre concluant sur l'opposé de ce littéral, est équivalent à la règle dont la partie prémisse est constituée par la réunion des parties prémisses des deux règles du couple, et concluant sur ⊥, symbole de l'absurde.

> On peut donc réécrire le système en remplaçant tous les couples de règles concluant l'une sur le littéral et l'autre sur son opposé, par une règle dont la partie prémisse est constituée par la réunion des parties prémisses des deux règles du couple, et concluant sur ⊥.

Par exemple:

Système initial	Système après réécriture
A1 ∧ A2 → B	A1 ∧ A2 ∧ A1 ∧ A4 ∧ A3 → ⊥
A1 ∧ A4 ∧ A3 → ¬B	A1 ∧ A2 ∧ A3 ∧ ¬A6 → ⊥
A6 → B	A6 ∧ A1 ∧ A4 ∧ A3 → ⊥
A3 ∧ ¬A6 → ¬B	A6 ∧ A3 ∧ ¬A6 → ⊥

---

[3] Remarques sur l'opération de coupure:
1°) Cette opération consiste à faire la **fermeture transitive** de la relation "→" (sur V).
2°) Cette fermeture peut générer une infinité de règles dans le cas où une règle a en conséquent un littéral apparaissant également dans sa partie prémisse
exemple: A partir de r1: E1 ∧ A → B, et r2: E2 ∧ B → A
On va générer r3: E1 ∧ E2 ∧ B → B qui peut se "fermer sur elle même".
On obtient r3': E1 ∧ E2 ∧ E1 ∧ E2 ∧ B → B puis
r3":E1 ∧ E2 ∧ E1 ∧ E2 ∧E1 ∧ E2 ∧ B → B etc...
qui sont en fait bien sur toujours la même règle r3.
Pour éviter le bouclage, on empêche la formation de règle de ce type.

A l'issue de cette deuxième opération, les littéraux V et ¬V n'apparaissent plus en conséquent de règle. Donc le mot déductible abs(V) a donc disparu du système.
Quand ces deux opérations ont pour but l'élimination d'un mot déductible M du système, on parlera de coupure sur M pour désigner la réunion de ces deux opérations.

c) Exemple

Pour compléter cette présentation, nous allons fournir, sans utiliser les notations précises qui seront définis dans la partie suivante, un exemple complet de fonctionnement de l'algorithme de MELOMIDIA sur un système formel.

Soit donc un SF du type décrit au chapitre 1 et ayant comme règles-axiome:

reg1: $A1 \rightarrow B$,
reg2: $\neg B \rightarrow C$,
reg3: $C \rightarrow D$,
reg4: $D \rightarrow E$,
reg5: $F \rightarrow \neg D$,
reg6: $A2 \wedge A3 \rightarrow G$
reg7: $A3 \wedge A8 \rightarrow H$
reg8: $A4 \wedge A6 \rightarrow \neg G$
reg9: $\neg A4 \wedge \neg A7 \rightarrow G$
reg10: $A8 \wedge G \rightarrow H$
reg11: $A9 \wedge \neg A8 \rightarrow \neg H$
reg12: $\neg A4 \wedge \neg A7 \wedge G \rightarrow G$
reg13: $A1 \wedge \neg A3 \rightarrow B$
reg14: $A9 \wedge \neg A9 \rightarrow \perp$

ce qui donne l'ensemble des mots déductibles: MD = {B, C, D, E, G, H} .

1°) Elimination des règles inutiles

a) la règle reg14: $A9 \wedge \neg A9 \rightarrow \perp$ est intrinsèquement contradictoire. Sa partie prémisse est indémontrable, sous réserve de BC-Cohérence ==> on l'élimine[4]

b) la règle reg12: $\neg A4 \wedge \neg A7 \wedge G \rightarrow G$ est inutile puisqu'elle conclue sur une de ses prémisses ==> on l'élimine[5]

c) la règle reg13: $A1 \wedge \neg A3 \rightarrow B$ est inutile puisqu'elle est moins générale que reg1: $A1 \rightarrow B$ ==> on l'élimine[6]

d) le mot B est déductible, sans que l'on ne déduise jamais ¬B. Par ailleurs, B n'est jamais présent en prémisse de règle

---

[4] une règle de ce type sera dite intrinsèquement illogique, et appelée une tautologie
[5] idem 4
[6] une règle de ce type sera dite subsumée, et appelée règle redondante

==> d.1) on peut éliminer la règle reg1: A1 → B concluant sur B: elle ne participera jamais à une démonstration contradictoire[7].

d.2) on peut éliminer la règle reg2: ¬B → C : le littéral ¬B est indémontrable, et n'est pas une hypothèse valide[8]

Il reste donc pour l'instant le système de règles-axiome:

reg3: $C \rightarrow D$,
reg4: $D \rightarrow E$,
reg5: $F \rightarrow \neg D$,
reg6: $A2 \wedge A3 \rightarrow G$
reg7: $A3 \wedge A8 \rightarrow H$
reg8: $A4 \wedge A6 \rightarrow \neg G$
reg9: $\neg A4 \wedge \neg A7 \rightarrow G$
reg10: $A8 \wedge G \rightarrow H$
reg11: $A9 \wedge \neg A8 \rightarrow \neg H$

e) Propagation de l'élimination de reg2: le mot C est déductible (il apparaissait en conséquent des règles-axiomes du système initial), mais plus aucune règle ne permet de le déduire

==> toutes les règles ayant C ou ¬C en prémisse ont des parties prémisses indémontrables, il faut les éliminer:

==> on élimine reg3: C → D (règle inatteignable)

f) Propagation de l'élimination de reg3: le littéral D n'est donc plus maintenant démontrable

==> f.1) la règle reg4: D → E a sa partie prémisse indémontrable, il faut l'éliminer (règle inatteignable)

f.2) le littéral ¬D est déductible, sans que l'on ne déduise jamais D. Par ailleurs, ¬D n'est jamais présent en prémisse de règle

==> on peut éliminer la règle reg5: F → ¬D, concluant sur ¬D: elle ne participera jamais à une démonstration contradictoire (règle annexe).

Il reste le système de règles-axiome:

reg6: $A2 \wedge A3 \rightarrow G$
reg7: $A3 \wedge A8 \rightarrow H$
reg8: $A4 \wedge A6 \rightarrow \neg G$
reg9: $\neg A4 \wedge \neg A7 \rightarrow G$
reg10: $A8 \wedge G \rightarrow H$
reg11: $A9 \wedge \neg A8 \rightarrow \neg H$

qui est minimal pour la recherche de démonstrations contradictoires.

2°) Opérations de coupure

---

[7] une règle de ce type sera appelée règle annexe

[8] une règle de ce type sera appelée règle inatteignable

- on commence par procéder à l'opération de coupure sur G, en opérant la fermeture transitive et en explicitant les contradictions sur G. On rajoute donc:

regFT(10,6)[9]: A8 ∧ A2 ∧ A3 → H

regFT(10,9): A8 ∧ ¬A4 ∧ ¬A7 → H

regCONTRAD(6,8)[10] : A2 ∧ A3 ∧ A4 ∧ A6 → ⊥

regCONTRAD(9,8) : ¬A4 ∧ ¬A7 ∧ A4 ∧ A6 → ⊥

tout en supprimant reg6, reg8, reg9 et reg10.

On élimine alors les règles inutiles de ce nouvel ensemble:

regFT(10,6): A8 ∧ A2 ∧ A3 → H est inutile, puisque moins générale que

reg7: A3 ∧ A8 → H

==> il faut l'éliminer (règle redondante)

et regCONTRAD(9,8) : ¬A4 ∧ ¬A7 ∧ A4 ∧ A6 → ⊥ a une partie prémisse contradictoire

==> il faut l'éliminer (règle intrinsèquement illogique)

Il reste:

reg7: A3 ∧ A8 → H

reg11: A9 ∧ ¬A8 → ¬H

regFT(10,9): A8 ∧ ¬A4 ∧ ¬A7 → H

reg CONTRAD(6,8) : A2 ∧ A3 ∧ A4 ∧ A6 → ⊥

- il reste à éliminer le mot H. L'opération de fermeture transitive est inutile, il suffit d'expliciter les contradictions. On rajoute donc:

regCONTRAD(7, 11) : A3 ∧ A8 ∧ A9 ∧ ¬A8 → ⊥

regCONTRAD(FT(10,9), 11) : A8 ∧ ¬A4 ∧ ¬A7 ∧ A9 ∧ ¬A8 → ⊥

Tout en éliminant reg7, reg11 et regFT(10,9).

On détecte alors que ces deux règles sont inutiles car ayant des parties prémisses contradictoires (règles intrinsèquement illogiques).

Il reste: regCONTRAD(6,8) : A2 ∧ A3 ∧ A4 ∧ A6 → ⊥

Cette règle contient la seule configuration d'hypothèses intiales conduisant à des déductions contradictoires.

d) Démonstrations syntaxiques et démonstrations sémantiques

Nous allons dans la partie suivante reprendre tout ces points, en présentant l'algorithme de vérification de la BC-Cohérence présent dans MELOMIDIA. Nous ferons simultanément la preuve syntaxique de sa justesse. Nous fournirons une description détaillée et, nous l'espérons, complète, des mécanismes et enchaînements constituant ce traitement.

---

[9] regFT(10,6) signifie: règle obtenue par Fermeture Transitive des règles 10 et 6.

[10] regCONTRAD(6,8) signifie: règle correspondant à l'explicitation de la contradiction entre reg6 et reg8.

La partie 3) reprendra les lemmes et théorèmes principaux de la partie 2) en leur associant de nouvelles démonstrations, basées cette fois sur une approche sémantique de la propriété de BC-Cohérence. Son objectif est à la fois de soulager le lecteur d'un examen minutieux de la partie 2) en lui fournissant des démonstrations plus simples des points délicats rencontrés plus haut, et surtout de donner l'esquisse d'une méthode générale de preuve d'algorithmes de vérification de la cohérence de systèmes d'ordre zéro plus.

2) L'algorithme de vérification de la BC-Cohérence utilisé par MELOMIDIA: Présentation et Preuve Syntaxique

Rappelons que nous nous intéressons ici à des SF du type décrit au 3) c) du chapitre 2. Pour simplifier l'écriture, on note BC l'union des axiomes issus des règles (BR) et des axiomes correspondant aux contraintes logique après réécriture (EC'), et on désignera par les mots "règles", "règles-axiome" ou "axiomes" indifféremment les éléments de BR ou de EC'.

On rappelle comment se définit le SF après réécriture, noté SF_{BC} :

1) Vocabulaire = $\{ \neg, \wedge, \rightarrow, MO_{BC} \} \cup \{\bot\}$

2) Règles de formation des phrases:
   - m et $\neg$m sont des phrases,appelées littéraux, si m est un mot courant
   - si p et q sont des littéraux ou des prémisses, p, q et $p \wedge q$ est une prémisse
   - si p est une prémisse et q un littéral ou $\bot$, $p \rightarrow q$ est une règle

3) Les Axiomes sont constitués d'un ensemble BC de règles

4) Les règles de déduction sont ($I\wedge$) et ($E\rightarrow$)

On rappelle que (cf 3)c) du chapitre 2)

- MP_{BC} ={Abs(p); $p \notin$ Cons(BC) et $p \wedge q \rightarrow \bot \in$ BC ou $p \wedge \neg q \rightarrow \bot \in$ BC et $q \in MD_{BC}$}
(ensemble des mots parasites)

- $MNR_{BC} = Abs(Prem(BC)) - MP_{BC}$
(ensemble des mots non récepteurs = ensemble des hypothèses initiales valides)

- $MD_{BC} = Abs(Cons(BC)) \cup MP_{BC}$
(ensemble des mots déductibles)

a) Elimination des règles-axiomes "témoins"

Le but de cette partie est de décrire des méthodes permettant d'éliminer du système formel le maximum de règles ne pouvant intervenir dans un processus de contradiction à partir d'hypothèses initiales valides. On va donc transformer un système de départ SD en système plus petit SA et montrer l'équivalence, en terme de cohérence, entre SA et SD.

Si on s'intéresse à la cohérence par rapport à un ensemble E tel que $E \cap MD_{SD} = \emptyset$ on sait, puisque SD $\supset$ SA, que $E \cap MD_{SA} = \emptyset$ également.

a.1) Elimination des règles intrinsèquement illogiques

Les premières règles à écarter du système sont les règles visiblement contradictoires ou inutiles.
On distingue:

- les règles insatifiables: ce sont les règles contenant une contradiction en partie prémisse: r : $A \rightarrow B$ et $\exists l$ tq $l \in A$ et $\neg l \in A$
- les "lapalissades": ce sont les règles concluant sur une de leur condition:

r : $A \rightarrow B$ et $B \in A$

Soit donc

ETAUT1[11](BC) = {r ∈ BC / ∃l ∈ MO_{BC} tq l ∈ Prem(r) et ¬l ∈ Prem(r)}.

ETAUT2[12](BC) = {r ∈ BC / ∃l ∈ MO_{BC} tq l ∈ Prem(r) et l ∈ Cons(r)}.

STAUT[13](BC) = BC — (ETAUT1(BC) ∪ ETAUT2(BC))

On démontre:

LEMME 1: **SF_{BC} cohérente** par rapport à E ($E \cap MD_{BC} = \emptyset$) ⇔ **$SF_{STAUT(BC)}$ cohérente** par rapport à E

Démonstration

⇒: Evident, si T est cohérent (par rapport à E), ∀T' tq $T \supset T'$, T' cohérent (par rapport à E)

⇐: $SF_{STAUT(BC)}$ est cohérent par rapport à E

donc ∀F tq $Valid_E(F)$ et ∀p, non(F |—STAUT(BC) $p \wedge \neg p$).

Donc ∀r ∈ ETAUT1(BC), non(F |— prem(r))

Donc ∀l, si non(F |—STAUT(BC) l) alors non(F |—STAUT(BC) ∪ ETAUT1(BC) l)

De même, ∀ r ∈ ETAUT2(BC), r de la forme $A \rightarrow B$ avec $B \in A$. Or
soit B n'est pas démontrable dans STAUT(BC) à partir de F, donc prem(r) n'est pas démontrable
soit B est démontrable dans STAUT(BC) à partir de F, auquel cas rajouter r à STAUT(BC) ne permettra pas de déduire un littéral nouveau

donc ∀l, si non(F |—STAUT(BC) l) alors non(F |—STAUT(BC) ∪ ETAUT2(BC) l)

donc ∀l, si non(F |—STAUT(BC) l) alors non(F |—STAUT(BC) ∪ ETAUT(BC) l)

donc SF_{BC} est cohérent par rapport à E.

---

[11] ETAUT1(BC) signifie: "Ensemble des tautologies de type 1".
Dans l'algorithme de Davis et Putnam, une tautologie est une clause contenant à la fois un littéral et son opposé. Rappelons qu'en Logique, la règle $A \Rightarrow B$ est équivalente à la clause $\neg A \vee B$.
Donc une tautologie peut provenir d'une règle de la forme $A \wedge B \Rightarrow A$ (tautologie de type 1)
ou $A \wedge B \wedge \neg B \Rightarrow C$ (tautologie de type 2).

[12] ETAUT2(BC) signifie: "Ensemble des tautologies de type 2" voir la note ci-dessus.

[13]STAUT signifie: "Sans tautologie"

a.2) Elimination des règles redondantes

Il s'agit de repérer les règles-axiomes qui n'interviendront pas dans une démonstration contradictoire à cause de l'existence dans le SF de règles plus générales, ie moins contraintes et de conclusion équivalente au regard de la cohérence. Trois cas d'"inclusions" d'une règle dans une autre règle vont être détaillés. On montre à chaque fois la validité de la suppression de la règle la moins générale.

LEMME 2: Soient deux règles r et r' d'un SF du type décrit au début de cette partie et telles que **Prem(r')** $\supset$ **Prem(r)** et
* Cas 1: **Cons(r) = Cons(r')**. Par exemple r: $A \rightarrow C$ et r': $A \wedge B \rightarrow C$,
* Cas 2: **Cons(r) =** $\perp$. Par exemple r: $A \rightarrow \perp$ et r': $A \wedge B \rightarrow C$,
* Cas 3: $\neg$**Cons(r)** $\in$ **Prem(r').** Par ex., r: $A \rightarrow B$ et r': $A \wedge \neg B \rightarrow C$,
alors SF_{BC} **cohérente** par rapport à E $\Leftrightarrow$ $SF_{BC - \{r'\}}$ **cohérente** par rapport à E

Démonstration

$\Rightarrow$: Evident, voir le lemme 1.

$\Leftarrow$:

Cas 1: Evident: l'axiome r' étant moins général que l'axiome r et permettant de déduire la même chose, il n'augmente en rien le "pouvoir de déduction" de BC — {r'}: si non(F $\vdash_{BC-\{r'\}}$ l) alors non(F $\vdash_{BC}$ l)
donc si $SF_{BC - \{r'\}}$ est cohérent par rapport à E, SF_{BC} l'est aussi.

Cas 2: Posons BC'. = BC — {r'}. Si $SF_{BC'}$ est cohérent par rapport à E, alors
$\forall$F tq $Valid_E(F)$, on sait que non(F $\vdash_{BC'}$ $\perp$)
$\Rightarrow$ non(F $\vdash_{BC'}$ Prem(r)) sinon (E$\rightarrow$) permettrait de déduire $\perp$ à partir de Prem(r) et r.
$\Rightarrow$ non(F $\vdash_{BC'}$ Prem(r')) car Prem(r') $\supset$ Prem(r).
Donc non(F $\vdash_{BC' \cup \{r'\}}$ Prem(r')) $\Rightarrow$ la règle r' ne peut être utilisée dans une démonstration utilisant les règles-axiomes de BC' $\cup$ {r'} , ie celles de BC.
Donc $\forall$l, si non(F $\vdash_{BC'}$ l), alors non(F $\vdash_{BC' \cup \{r'\}}$ l).
Donc si $SF_{BC'}$ est cohérent par rapport à E, SF_{BC} l'est aussi.

Cas 3: Posons BC' = BC — {r'}. Quelque soit F tel que $Valid_E(F)$,
- soit F $\vdash_{BC'}$ Prem(r) $\Rightarrow$ F $\vdash_{BC'}$ Cons(r)
$\Rightarrow$ non(F $\vdash_{BC'}$ $\neg$Cons(r)) puisque $SF_{BC'}$ est cohérent
$\Rightarrow$ non(F $\vdash_{BC'}$ Prem(r')), d'où non(F $\vdash_{BC}$ Prem(r'))
- soit non(F $\vdash_{BC'}$ Prem(r))
$\Rightarrow$ non(F $\vdash_{BC'}$ Prem(r')) puisque Prem(r') $\supset$ Prem(r),
$\Rightarrow$ non(F $\vdash_{BC}$ Prem(r')).
Là encore, la règle r' ne peut être utilisée dans une démonstration utilisant les règles-axiomes de BC. Donc $\forall$l, si non(F $\vdash_{BC'}$ l), alors non(F $\vdash_{BC}$ l).

Donc si $SF_{BC'}$ est cohérent par rapport à E, SF_{BC} l'est aussi.

Soit donc ERSUB et SSUB les ensembles suivants:

ERSUB[14](BC)={ r ∈ BC / ∃ r' ∈ BC tq Prem(r) ⊃ Prem(r')
et (Cons(r) = Cons(r') ou Cons(r') = ⊥ ou ¬Cons(r') ∈ Prem(r))}

SSUB[15](BC) = BC — ERSUB(BC)

On a le résultat:

LEMME 3: **SF_{BC} cohérente** par rapport à E ⇔ **$SF_{SSUB(BC)}$ cohérente** par rapport à E

Démonstration évidente d'après le lemme 2

a.3) Elimination des règles annexes et des règles inatteignables

a.3.1) Définitions

On note, pour un système formel d'axiomes BC et pour tout littéral m tel que Abs(m) appartient à MD_{BC}:

BC(Prem, m) = {r ∈ BC / m ∈ Prem(r)},

BC(Prem,"m") = BC(Prem,Abs(m)) ∪ BC(Prem,¬Abs(m))

BC(Cons, m) = {r ∈ BC / m ∈ Cons(r)}

BC(Cons,"m") = BC(Cons,Abs(m)) ∪ BC(Cons,¬Abs(m))

BC("m") = BC(Prem,"m") ∪ BC(Cons,"m")

a.3.2) Opération de Base

Nous allons maintenant nous intéresser aux règles-axiomes du système formel ne pouvant intervenir dans la démonstration d'une contradiction. Ces règles se divisent en deux groupes:
- Celles qui, par le littéral qu'elles déduisent, ne saurait intervenir dans une démonstration contradictoire (règles annexes).
- Celles dont les conditions qu'elles imposent en partie prémisse ne pourront jamais être satisfaites. En effet, elles contiennent en partie prémisse un littéral ne pouvant faire partie des hypothèses initiales et ne pouvant être déduit (règles inatteignables).

Soit donc BC une base de règles constituant les axiomes d'un SF du type décrit au début de cette partie, et m un mot différent de ⊥ et appartenant au SF.

---

[14] ERSUB(BC) signifie: "Ensemble des règles subsumées". En Logique, une expression e1 est subsumée par une expression e2 si e1 est incluse dans e2;

[15] SSUB signifie: "Sans subsomption"

On appelle ERANN[16](BC,m), ERIND[17](BC,m) et ERINU[18](BC,m), les sous-ensembles de BC définis par:

ERANN(BC,m) = {r ∈ BC tq Abs(Cons(r)) = m
et non(∃r' ∈ BC tq Cons(r') = ¬Cons(r))
et non(∃r'' ∈ BC tq Cons(r) ∈ Prem(r''))}

ERIND(BC,m) = {r ∈ BC / ∃p ∈ Prem(r) tq Abs(p) = m
et non(∃r' ∈ BC tq Cons(r') = p)}

ERINU(BC,m) = ERANN(BC,m) ∪ ERIND(BC,m)

Le premier ensemble correspond aux règles annexes concluant sur m, le deuxième correspond aux règles qui n'apparaîtront jamais dans une démonstration si m n'appartient pas à l'ensemble des hypothèses initiales valides.

Plus concrètement, si on dispose pour chaque mot de BC de quatre indicateurs appelés respectivement POSC(m), NEGC(m), POSP(m) et NEGP(m) indiquant la présence (valeur 1) ou l'absence (valeur 0) du mot m respectivement:
- POSitivement en Conséquent d'au moins une règle de BC (il existe une règle de BC qui conclue sur m),
- NEGativement en Conséquent d'au moins une règle de BC (il existe une règle de BC qui conclue sur ¬m),
- POSitivement en Prémisse d'au moins une règle de BC (il existe une règle contenant m en partie prémisse),
- NEGativement en Prémisse d'au moins une règle de BC (il existe une règle contenant ¬m en partie prémisse),

on peut dresser le tableau suivant:

POSC(m)	NEGC(m)	POSP(m)	NEGP(m)	ERINU(BC,m)
0	0	0 ou 1	0 ou 1	BC("m")
1	0	0	0 ou 1	BC("m")
1	0	1	0 ou 1	BC(Prem,¬m)
0	1	0 ou 1	0	BC("m")
0	1	0 ou 1	1	BC(Prem,m)
1	1	0 ou 1	0 ou 1	∅

---

[16] ERANN(BC,m) signifie: "Ensemble des règles annexes de BC contenant m en conséquent"

[17] ERIND(BC,m) signifie: "Ensemble des règles indéclenchables de BC à cause de m"

[18] ERINU(BC,m) signifie: "Ensemble des règles inutiles de BC à cause de m".
Cette notion est à rapprocher du concept de littéral pur donnée par Davis et Putnam. Il s'agit en effet d'un littéral présent dans un ensemble de clauses sans que son opposé y soit:
l est pur si l ∈ S et Opposé(l) ∉ S
Si maintenant on transforme la règle A → B en la formule ¬A ∨ B*, et que l'on considère l'ensemble des clauses contenant un littéral pur redéfini comme suit: l est pur ssi
[l* ∈ S et Opposé(l) ∉ S et (Opposé(l))* ∉ S] ou [l ∈ S et (Opposé(l))* ∉ S],
on retrouve les deux ensembles ERANN et ERIND.

a.3.3) Itération, Algorithme

On note **SRINU[19](BC, MD0)** le sous-ensemble maximal de BC défini par:
$\forall m \in$ **MD0, ERINU(SRINU(BC, MD0), m)** = $\emptyset$. Autrement dit:
$\forall r \in$ SRINU(BC,MD0), $\forall m \in$ MD0, $r \notin$ ERINU(SRINU(BC,MD0), m)
On va montrer une méthode de calcul de SRINU(BC, MD0), puis prouver l'équivalence, au plan de la cohérence, entre SF_{BC} et $SF_{SRINU(BC, MD0)}$

Concrètement, MD0 désignera toujours l'ensemble des mots déductibles du SF dont on étudie la BC-Cohérence. La construction de SRINU(BC, MD0) repose sur un algorithme supposant que l'on dispose d'une liste des éléments de MD0. Cet algorithme construit deux suites (b_i) et (l_i) où b_i désigne une base de règles et l_i un mot de MD0, puis définit SRINU(BC,MD0) comme la limite des (b_i) :

Algorithme "Construction du sous-ens. max. de BC utile pour la cohérence"
$b_0 \Leftarrow BC$, $i \Leftarrow 0$
$m \Leftarrow$ Premier_Elément_De(MD0)
TANT QUE m <> NIL
Calculer ERINU(b_i, m)
SI ERINU(b_i, m) = $\emptyset$
ALORS $m \Leftarrow$ Elément_Suivant_De(MD0)
SINON $b_{i+1} \Leftarrow b_i$ — ERINU(b_i, m)
$l_i \Leftarrow m$
$m \Leftarrow$ Premier_Elément_De(MD0)
$i \Leftarrow i + 1$
FIN_TANT_QUE
SRINU(b_0, MD0) $\Leftarrow b_i$

Attention: Ne pas confondre MD0 et MD_{b0}, il ne sont pas a priori identiques

Montrons que cet algorithme calcule bien SRINU(BC,MD0).
1°) Il est évident que l'ensemble obtenu grâce à l'algorithme vérifie la propriété:
$\forall m \in$ MD0, ERINU(SRINU(BC, MD0), m) = $\emptyset$ (m ne prend la valeur NIL qu'après avoir parcouru toute la liste des éléments de MD0).
2°) Toutefois, est-il LE plus grand sous-ensemble de BC vérifiant cette propriété? L'algorithme opérant par retraits successifs des règles ne vérifiant pas la propriété, il est clair que l'on obtient UN "plus grand élément" de l'ensemble des bases de règles sans règles inutiles. On va maintenant montrer que ce plus grand élément est unique, ie indépendant de l'ordre dans lequel sont effectués les retraits dans la base
de règles BC. On pourra alors parler **DU** s.-e. **maximal** de BC vérifiant la propriété.

---

[19] SRINU(BC, MD0) signifie "Sans Règles de BC Inutiles à cause d'éléments de MD0".

LEMME 4: SRINU(BC, MD0) ne dépend pas de l'ordre dans lequel apparaissent les entités dans la liste des entités déductibles

Autrement dit,

Si (b_i^1) et (b_i^2) sont deux suites de bases de règles associées aux deux suites de mots (l_i^1) et (l_i^2) issues de l'algorithme en s'appuyant sur deux listes contenant les éléments de MD0 classées avec deux ordres différents. C'est-à-dire si:

$b_0^1 = b_0^2 = BC$ et $b_{i+1}^j = b_i^j$ — ERINU(b_i^j, l_i^j) avec j = 1 et j = 2

on a $\mathbf{lim(b_i^1) = lim(b_i^2)}$

Démonstration

1°) Ces limites existent. En effet, l'algorithme se termine puisqu'il ne peut que soustrair des éléments d'un ensemble fini de règles, donc il existe n_1 et n_2 tq

$\forall m \in MD0$, $\mathbf{ERINU(b_{n1}^1, m) = \varnothing}$ et $\mathbf{ERINU(b_{n2}^2, m) = \varnothing}$,

2°) Supposons qu'il existe une règle, notée r1 qui appartiennent à b_{n1}^1 mais qui n'appartienne pas à b_{n1}^2, ie $r1 \in b_{n1}^1 - b_{n1}^2$.

-> Si $r1 \in b_{n1}^1$, cela implique que $\forall m \in MD0$, $r1 \notin ERINU(b_{n1}^1, m)$

Donc A) il existe une règle, notée r_2^0 tq

$Cons(r_2^0) = \neg Cons(r1)$ ou $Cons(r1) \in Prem(r_2^0)$

B) si $Card\{p \in Prem(r1)$ tq $Abs(p) \in MD0\} = p1$

alors il existe au moins p1 règles, notées $r_2^1, r_2^2, ..., r_2^{p1} \in (b_{n1}^1)^{p1}$ tq

$\forall p / p \in Prem(r1)$ et $Abs(p) \in MD0$, $\exists i \leq p1$ tq $Cons(r_2^i) = p$

On appelera Justification de r1 dans b_{n1}^1 et on notera Justif(r1) l'ensemble:

$Justif(r1) = \{r_2^0, ..., r_2^{p1}\}$.

-> Si r1 n'appartient pas à b_{n2}^2, alors il existe $i < n2$ tq $r1 \in ERINU(b_i^2, l_i^2)$

Or Justif(r1) est inclus dans b_0^2, donc pour que r1 puisse appartenir à $ERINU(b_i^2, l_i^2)$, c'est qu'il existe $j \leq p1$ tq $r_2^j \in ERINU(b_{kj}^2, l_{kj}^2)$ avec $kj \leq i$

Soit r_2^{j0} le premier élément de Justif(r1) à disparaître, ie

$\forall r \in Justif(r1)$, $r \in b_{kj0}$, et $r_2^{j0} \in ERINU(b_{kj0}^2, l_{kj0}^2)$

Alors, par définition, $r1 \notin ERINU(b_{kj0}^2, l_{kj0}^2)$, donc $\mathbf{k_{j0} < i}$

Donc, $r1 \in b_{n1}^1$ et $r1 \notin b_{n2}^2 \Rightarrow \exists r_2^{j0} / r_2^{j0} \in b_{n1}^1$ et $r_2^{j0} \notin b_{kj0}^2$ avec $k_{j0} < i \leq n2$

On répète le même raisonnement itérativement. On obtient:

$r_2^{j0} \in b_{n1}^1$ et $r_2^{j0} \notin b_{kj0}^2 \Rightarrow \exists r_3^{j0} / r_3^{j0} \in b_{n1}^1$ et $r_3^{j0} \notin b_{lj0}^2$ avec $l_{j0} < k_{j0}$

$\Rightarrow$ ...

$\Rightarrow \exists r_z^{j0} / r_z^{j0} \in b_{n1}^1$ et $r_z^{j0} \notin b_0^2$ ce qui est impossible.

Cette démonstration pouvant évidemment être réécrite pour démontrer que $b_{n2}^2 - b_{n1}^1 = \emptyset$, on a donc que $b_{n1}^1 = b_{n2}^2$

Donc l'ensemble obtenu est maximal et ne dépend pas de l'ordre des mots déductibles dans la liste.

Il reste bien sur à démontrer l'équivalence, au plan de la cohérence entre SF_{BC} et $SF_{SRINU(BC, MD0)}$. On va pour cela démontrer le lemme suivant:

LEMME 5: **si E est tel que $E \cap MD_B = \emptyset$,**
SF_B cohérent par rapport à E $\Leftrightarrow$ $\forall m \in MO_B - E$, $SF_{\{B - ERINU(B,m)\}}$ cohérent par rapport à E

Démonstration

$\Rightarrow$: Evident, voir lemme 1

$\Leftarrow$: Supposons SF_B incohérent par rapport à E, posons

B' = B — ERINU(B, m) avec $m \in MO_B$ — E quelconque,

montrons que $SF_{B'}$ est incohérent par rapport à E

SF_B incohérent équivaut à: il existe $H = \{h_0, \ldots, h_n\}$ tq $VALID_E(H)$ tq:

- soit $h_0, \ldots, h_n \vdash_B Q$ et $h_0, \ldots, h_n \vdash_B \neg Q$
- soit $h_0, \ldots, h_n \vdash_B \perp$

c'est-à-dire

- soit il existe deux suites de phrases $p_1^1, p_2^1, \ldots, p_{n1}^1$ avec $p_{n1}^1 = Q$
  $p_1^2, p_2^2, \ldots, p_{n2}^2$ avec $p_{n2}^2 = \neg Q$
- soit il existe une suite de phrases $p_1^3, p_2^3, \ldots, p_{n3}^3$ avec $p_{n3}^3 = \perp$

Telles que quels que soient $j \leq 3$ et $i \leq n_j$

- soit p_i^j est une hypothèse: $p_i^j = h_k$ ou
- soit p_i^j est un axiome du SF, ie une règle de B,

et dans ce cas ou sait que $\exists k < i$ tq $\quad \dfrac{p_k^j \quad p_i^j}{p_{i+1}^j}$ (E→)

- soit p_i^j est tel que $p_1^j, \ldots, p_{i-1}^j \vdash p_i^j$

La démonstration consiste à montrer que toute phrase p_i^j démontrable dans SF_B (à partir de E) l'est également dans $SF_{B'}$ (à partir de E).

Dans la suite, (s_i) désignera indifféremment l'une des trois familles (p_i^1), (p_i^2) ou (p_i^3). Démontrons **par récurrence que (s_i) est démontrable dans $SF_{B'}$.**

* s_1 est démontrable dans $SF_{B'}$. En effet, les seules déductions possibles sont (E→) $\frac{s1: A \quad s2: A \to B}{s3: B}$ et (E∧) $\frac{s1: A \quad s2: B}{s3: A \wedge B}$ donc s1 est nécessairement une hypothèse initiale, donc est démontrable dans $SF_{B'}$.

* supposons les s_i , $i \leq k$ démontrables dans $SF_{B'}$, montrons que s_{k+1} est démontrable dans $SF_{B'}$. Trois cas peuvent se produire:

- soit $s_{k+1} = h_l$ ($l \leq n$), d'où s_{k+1} est évidemment démontrable dans $SF_{B'}$
- soit s_{k+1} est tel que $s_1, \ldots, s_k \vdash_B s_{k+1}$. Or, par hypothèse de récurrence, chaque s_i ($i \leq k$) est démontrable dans $SF_{B'}$, et **les règles de déduction de SF_B sont les mêmes que celles de $SF_{B'}$,**

donc $s_1, \ldots, s_k \vdash_{B'} s_{k+1}$

- soit s_{k+1} est une règle-axiome de SF_B. Alors on sait qu'il existe $l < k+1$ tq:

(E→): $\frac{s_l \qquad s_{k+1}}{s_{k+2}}$ et s_l est démontrable dans $SF_{B'}$.

La question est de savoir si **la règle (notons-la R) repérée par s_{k+1} appartient à B'**, ie vérifier que: **$\forall m \in MO_B - E$, $R \notin ERINU(B, m)$.**

* tout d'abord, on sait que s_l = Prem(R) est démontrable dans SF_B, donc: $\forall p \in$ Prem(R), p est démontrable dans SF_B à partir de E.

Or si Abs(p) $\in MO_B - E$, Abs(p) $\notin$ E, donc p est inféré (il ne peut pas faire partie des hypothèses initiales). Or, seule (E→) infère des littéraux. Donc il existe $j < l$ tel que s_j désigne une règle R' tq p = Cons(R'), et que l'on a l'inférence

(E→): $\frac{s_{l'} \qquad sj}{s_{j+1} = p}$ avec $l' \leq j$

Donc $\forall p \in$ Prem(R) tq Abs(p) $\in MO_B - E$, $\exists R' \in B$ tel que $p \in$ Cons(R')

donc $\forall m \in MO_B - E$, $R \notin$ ERIND(B, m)

* il reste à montrer que $\forall m \in MO_B - E$, $R \notin ERANN(B, m)$. On va se servir pour cela de l'existence de s_{k+2} "dans" B. En effet,

+ soit $s_{k+2} = \perp$ ((s_i) désigne la famille (p_i^3)). Dans ce cas on a immédiatement que $R \notin ERANN(B,m)$ puisque $\perp \notin MD_B$ donc $ERANN(B, \perp)$ n'existe pas donc $\forall m$, $R \notin ERANN(B,m)$.

+ soit $s_{k+2} = Q$ (resp. $\neg Q$ si (s_i) désigne la famille (p_i^2))
ie $Cons(R) = Q$ (resp. $\neg Q$). Or
1°) il existe une démonstration de $\neg Q$ (resp. Q) dans SF_B à partir de H,
2°) $Q \in MD_B$ ($Q \in Abs(Cons(R))$) donc H ne peut contenir $\neg Q$ (resp. Q)
puisque $E \supset Abs(H)$ et $E \cap MD_B = \emptyset$ donc $H \cap MD_B = \emptyset$,
donc **on sait qu'il existe R' appartenant à B tel que Cons(R') = $\neg Q$** (resp. Q) , donc $\forall m$, $R \notin ERANN(B,m)$.

+ soit $s_{k+2} <> Q$ (resp. $\neg Q$ et $\perp$), ie s_{k+2} est un littéral intermédiaire dans la déduction de Q (resp. $\neg Q$ et $\perp$). Il doit nécessairement être utilisé ultérieurement dans une inférence. Donc il existe l1 tel que
(D1): $\dfrac{s_{k+2} \qquad s_{l1}}{s_{l2}}$ avec $l1 > k+2$ et $l2 > l1$ et:
soit (D1) = (E→) ie s_{l1} est une règle notée R" et $s_{k+2} = Prem(R")$
soit (D1) = (I∧). Dans ce cas, s_{l2} est une conjonction de littéraux, donc est différent de Q (resp. $\neg Q$ et $\perp$), donc il existe $l3 > l2$ et $l4 > l3$ tq
(D2): $\dfrac{s_{l2} \qquad s_{l3}}{s_{l4}}$ et on peut faire le même raisonnement sur (D2) que sur (D1).
Le nombre d'étapes de la démonstration étant fini, et cette démonstration aboutissant au littéral Q (resp. $\neg Q$ et $\perp$), on a donc nécessairement qu'il existe i tel que (Di) = (E→) après déduction de s_{k+2}.
Supposons que cette inférence ait lieu i_0 inférences après avoir déduit s_{k+2},
ie $(Di_0) = (E\rightarrow)$. Notons R" la règle concernée. On a:
$sl2i_0-2 = Prem(R")$, $sl2i_0-1 = R"$ et $sl2i_0+1 = Cons(R")$. Or, $sl2i_0-2$ est obtenue à partir d
s_{k+2} en utilisant seulement (I∧) $\Rightarrow s_{k+2} \in sl2i_0-2$ ie $s_{k+2} \in Prem(R")$.
Et $s_{k+2} = Cons(R)$, donc **Cons(R) ∈ Prem(R")**, donc $\forall m$, $R \notin ERANN(B, m)$

donc $\forall m \in MO_B - E$, $R \notin ERINU(B, m)$, ie $R \in B'$
donc s_{k+1} est démontrable dans $SF_{B'}$

L'hypothèse de récurrence est donc vérifiée. Donc quelque soient i et j, p_i^j est démontrable dans $SF_{B'}$, donc **SF_B, est incohérente par rapport à E.**

On peut maintenant énoncer le théorème:

THEOREME 1:
Si E est tel que $E \cap MD_B = \emptyset$,
SF_B cohérent par rapport à $E \Leftrightarrow \forall F$ tel que $MO_B - E \supset F$,
$SF_{SRINU(B, F)}$ cohérent par rapport à E

[En pratique, l'ensemble E qui nous intéresse est MNR0, l'ensemble des mots non récepteurs du système dont on cherche la BC-Cohérence, et l'ensemble F correspond à l'ensemble MD0 des mots déductibles de ce même système.]

Démonstration

$\Rightarrow$: Evident

$\Leftarrow$: On sait qu'on peut calculer SRINU(B, F) grâce à l'algorithme,
ie définir SRINU(B, F) comme la limite d'une suite (b_i) finie de bases de règles définie par $b_0 = B$ et $b_{i+1} = b_i$ — ERINU(b_i, l_i), avec l_i le premier littéral d'une liste quelconque des éléments de F tq ERINU(b_i, l_i) $\neq \emptyset$
On peut donc poser: **SRINU(B, F) = b_n = b_{n-1} — ERINU(b_{n-1} , l_{n-1})**
avec $l_{n-1} \in F$ ie $l_{n-1} \in MO_{b_{n-1}} - E$ (car $F \cap E = \emptyset$ et
si $l_{n-1} \notin MO_{b_{n-1}}$ alors ERINU(b_{n-1} , l_{n-1}) = $\emptyset$)
De plus, $B \supset b_{n-1}$ donc $MD_b \supset MD_{b_{n-1}}$ **donc** $MD_{b_{n-1}} \cap E = \emptyset$
Donc la proposition $SF_{SRINU(B, F)}$ cohérent par rapport à E s'écrit:
$E \cap MD_{b_{n-1}} = \emptyset$ et $SF_{\{b_{n-1} - ERINU(b_{n-1}, l_{n-1})\}}$ cohérent par rapport à E ,
avec $l_{n-1} \in MO_{b_{n-1}} - E$
donc (lemme 5) on a: (I): $SF_{b_{n-1}}$ cohérent par rapport à E.

En fait, quel que soit i, on a $B \supset b_i$ donc $MD_{b_i} \cap E = \emptyset$
et $l_i \in F$ et ERINU(b_i , l_i) = $\emptyset$ donc $l_i \in MO_{b_i} - E$ et $b_i = b_{i-1}$ — ERINU(b_{i-1} , l_{i-1})
Donc quel que soit i, la proposition SF_{b_i} cohérent par rapport à E peut s'écrire:
$E \cap MD_{b_{i-1}} = \emptyset$ et $SF_{\{b_{i-1} - ERINU(b_{i-1}, l_{i-1})\}}$ cohérent par rapport à E
avec $l_{i-1} \in MO_{b_{i-1}} - E$
ie $SF_{b_{i-1}}$ cohérent par rapport à E .
On arrive donc, de proche en proche, à SF_{b_0} ie SF_B cohérent par rapport à E. CQFD.

a.3.4) Exemple

BC = { $A \rightarrow B$, $\neg B \rightarrow C$, $C \rightarrow D$, $D \rightarrow E$,
$F \rightarrow \neg D$, $G \rightarrow I$, $H \rightarrow \neg I$}

donc MD_{BC} = MD0 = {B, C, D, E, I} . Soit (B, C, D, E, I) la liste correspondante

b0 = {$A \rightarrow B$, $\neg B \rightarrow C$, $C \rightarrow D$, $D \rightarrow E$, $F \rightarrow \neg D$, $G \rightarrow I$, $H \rightarrow \neg I$}
ERINU(b0, B) = {$A \rightarrow B$} $\cup$ {$\neg B \rightarrow C$}

donc b1 = {$C \rightarrow D$, $D \rightarrow E$, $F \rightarrow \neg D$, $G \rightarrow I$, $H \rightarrow \neg I$}
ERINU(b1, B) = $\emptyset$, ERINU(b1, C) = {$C \rightarrow D$}

donc b2 = {$D \rightarrow E$, $F \rightarrow \neg D$, $G \rightarrow I$, $H \rightarrow \neg I$}
ERINU(b2, B) = ERINU(b2, C) = $\emptyset$,
ERINU(b2, D) = {$D \rightarrow E$, $F \rightarrow \neg D$}

donc b3 = {$G \rightarrow I$, $H \rightarrow \neg I$}
ERINU(b3, B) = ERINU(b3, C) = ERINU(b3, D) = ERINU(b3, E) = ERINU(b3, I) = $\emptyset$,

donc SRINU(BC, MD_{BC}) = {$G \rightarrow I$, $H \rightarrow \neg I$}

b) Recherche des contradictions: Coupure sur un mot déductible

Les opérations qui vont maintenant être décrites reprennent l'opération de coupure esquissée au 1)b). Il s'agit d'opérer la fermeture transitive de la relation $\rightarrow$, puis d'expliciter les contradictions potentielles sur le mot que l'on cherche à éliminer.

b.1) Réécriture des règles intrinsèquement contradictoires

Il s'agit ici de finir de débarrasser la base de règles de règles possédant en conséquent un littéral contradictoire avec une des prémisses. On définit donc:

$BC(Contrad) = \{r \in BC / \neg Cons(r) \in Prem(r)\}$

Les règles de cet ensemble sont donc de la formé: $A \wedge l \rightarrow \neg l$ ou $A \wedge \neg l \rightarrow l$

On va les réécrire respectivement $A \wedge l \rightarrow \bot$ et $A \wedge \neg l \rightarrow \bot$.

On pose donc:

$BC(\bot) = \{r: Prem(r') \rightarrow \bot$, avec $r' \in BC(Contrad)\}$.

et **$SRIC^{20}(BC) = (BC - BC(Contrad)) \cup BC(\bot)$**

On va démontrer:

LEMME 6:

SF_{BC} cohérent par rapport à E et $E \cap MD_{BC} = \emptyset$ $\Leftrightarrow$ $SF_{SRIC(BC)}$ cohérent par rapport à E

Démonstration

SF_{BC} cohérent par rapport à E

$\Leftrightarrow \forall F$ tq $Valid_E(F)$, non($F \vdash_{BC} p \wedge \neg p$) et non($F \vdash_{BC} \bot$)

$\Leftrightarrow \forall F$ tq $Valid_E(F)$, d'une part: $\forall r \in BC(Contrad)$, non($F \vdash_{BC} Prem(r)$)

sinon $F \vdash_{BC} \neg l$ bien que $F \vdash_{BC} l$ puisque $F \vdash_{BC} Prem(r)$,

et d'autre part: non($F \vdash_{BC - BC(Contrad)} p \wedge \neg p$) et non($F \vdash_{BC - BC(Contrad)} \bot$)

$\Leftrightarrow \forall F$ tq $Valid_E(F)$, d'une part: $\forall r \in BC(\bot)$, non($F \vdash_{BC} Prem(r)$)

et d'autre part: non($F \vdash_{BC - BC(Contrad)} p \wedge \neg p$) et non($F \vdash_{BC - BC(Contrad)} \bot$)

$\Leftrightarrow \forall F$ tq $Valid_E(F)$, d'une part: $\forall r \in BC(\bot)$, non($F \vdash_{SRIC(BC)} Prem(r)$)

et d'autre part: non($F \vdash_{BC - BC(Contrad)} p \wedge \neg p$) et non($F \vdash_{BC - BC(Contrad)} \bot$)

$\Leftrightarrow \forall F$ tq $Valid_E(F)$, non($F \vdash_{SRIC(BC)} p \wedge \neg p$) et non($F \vdash_{SRIC(BC)} \bot$)

$\Leftrightarrow SF_{SRIC(BC)}$ cohérent par rapport à E.

Remarque: $MD_{BC} \supset MD_{SRIC(BC)}$ donc $E \cap MD_{BC} = \emptyset \Rightarrow E \cap MD_{SRIC(BC)} = \emptyset$

---

[20] SRIC(BC, "m") signifie:" Sans règle intrinsèquement contradictoire".

b.2) Fermeture Transitive

* On désigne par **r: A ∪ (B — {l}) → l'**
où A et B désignent des prémisses, l un littéral et l' un littéral ou ⊥, la règle définie par: p ∈ Prem(r) ssi p ∈ A ou p ∈ B — {l}, et Cons(r) = l'.
* On note, pour tout littéral m tel que Abs(m) appartient à MD_{BC} :

Res(BC, Prem-Cons, m) = {r: Prem(r') ∪ (Prem(r") — {m}) → Cons(r"),
avec r' ∈ BC(Cons, m) et r" ∈ BC(Prem, m)}
Res(BC, Prem-Cons,"m") = Res(BC, Prem-Cons, Abs(m))
∪ Res(BC, Prem-Cons, ¬Abs(m))

Et on définit l'opération de fermeture transitive de "→" sur m dans BC par:
FT(BC, m) = (BC — BC(Prem,"m")) ∪ Res(BC, Prem-Cons,"m")
ie la base de règles initiale sans les règles ayant m ou ¬m en prémisse mais avec toutes les règles obtenues par fermeture transitive de "→" sur m.

Exemple:
BC = {A → B, A' → B, B ∧ C → D, B ∧ E → F}
FT(BC,B) = {A → B, A' → B, A ∧ C → D, A'∧ C → D, A ∧ E → F, A'∧ E → F}

On va montrer que le système obtenu après l'opération de fermeture transitive sur m, a, sous certaines conditions, les mêmes capacités de déduction que le système initial:

LEMME 7: Soit BC les axiomes d'un SF du type décrit au début du 3) et :
Si ETAUT(BC) = ∅, BC(Contrad) = ∅ et m ∈ MD_{BC} alors
∀F tq Abs(F) ∩ MD_{BC} = ∅, ∀l ∈ MO_{BC} , F ⊢$_{BC}$ l ⇔ F ⊢$_{FT(BC,m)}$ l

Démonstration
⇒: Soit (s_i) une démonstration de l dans SF_{BC}.
On va montrer **par récurrence** que **∀s_i tq s_i ∈ MO_{BC} , s_i est démontrable dans $SF_{FT(BC, m)}$**.
En effet:
- $s_1 \in MO_{BC}$ et s_1 est démontrable dans $SF_{FT(BC, m)}$.
- supposons la propriété vraie jusqu'au rang j, ie
∀k ≤ j et si $s_k \in MO_{BC}$, alors s_k est démontrable dans $SF_{FT(BC, m)}$ et $s_{j+1} \in MO_{BC}$
montrons que s_{j+1} est démontrable dans FT(BC, m).

En effet, soit s_{j+1} est une hypothèse initiale $f_i \in F$, donc s_{j+1} est démontrable dans FT(BC, m), soit s_{j+1} est inféré. Dans ce cas on sait qu'il existe forcément la dérivation: s_i s_j

------- (E→) avec s_j une règle de BC notée R1, s_{j+1} = Cons(R1)

s_{j+1}

et s_i = Prem(R1) , i < j , s_i s'écrivant : $l_1 \wedge l_2 \wedge ... \wedge l_{n1}$

> Tout d'abord, s_i est démontrable dans SF_{BC}, donc

$\forall i, i \leq n1, \exists k < i$ tq $l_i = s_k$ donc **$\forall i \leq n1$, l_i est démontrable dans $SF_{FT(BC,m)}$.**

> Maintenant,

- soit s_j (ie R1) $\in$ FT(BC, m), auquel cas s_{j+1} est trivialement démontrable dans $SF_{FT(BC, m)}$,
- soit R1 $\notin$ FT(BC, m), ie R1 $\in$ BC(Prem,"m").

Dans ce cas, on sait que $m \in Abs(s_i)$. Donc il existe un sous-ensemble de s_i composé de littéraux tous égaux à m ou un sous-ensemble de s_i composé de littéraux tous égaux à ¬m (le "ou" est exclusif puisque ETAUT(BC) = ∅) .

Supposons que cet ensemble soit composé des k_0 premiers littéraux de s_i:

$\forall i \leq k_0$ et $\forall j \leq k_0$, $l_i = l_j$ et $l_i = lit_m$ avec lit_m = m ou (exclusif) $lit_m = \neg m$

Or on sait que $m \in MD_{BC}$ donc nécessairement il existe g et h tels que:

s_g s_h

---------(E→) avec s_h une règle, notée R2, de BC tq s_{h+1} = Cons(R2) = lit_m

s_{h+1}

et **s_g** = Prem(s_h) = **Prem(R2)**.

De plus, on sait que $m \notin s_g$ puisque ETAUT(BC) ∪ BC(Contrad) = ∅.

De même que pour s_i, $\forall l \in s_g \exists k < g$ tel que $l = s_k$, donc l est démontrable dans $SF_{FT(BC,m)}$ donc **s_g est démontrable dans $SF_{FT(BC,m)}$**

Or R2 $\in$ BC(Cons,lit_m) et R1 $\in$ BC(Prem,lit_m) avec lit_m = m ou $lit_m = \neg m$

donc il existe **R $\in$ FT(BC,m)** tel que

Prem(R) = Prem(R2) ∪ (Prem(R1) — {l}) = **$s_g \wedge l_{k_0+1} \wedge ... \wedge l_{n1}$**

Cons(R) = Cons(R1) = **s_{j+1}**

> On a vu que: $\forall i \leq n1$, l_i est démontrable dans $SF_{FT(BC,m)}$, et s_g est démontrable dans $SF_{FT(BC,m)}$, donc ((I∧)) **Prem(R) est démontrable dans $SF_{FT(BC,m)}$, donc Cons(R) = s_{j+1} est démontrable dans $SF_{FT(BC,m)}$.**

L'hypothèse de récurrence est bien vérifiée. La propriété est établie.

$\Leftarrow$: La démonstration est du même type que celle ci-dessus, mais plus simple. La seule difficulté, dans la récurrence, consiste là aussi à montrer que

s_{j+1} = Cons(s_j) est démontrable dans SF_{BC}, même si $s_j \notin$ BC:

Dans ce cas, on sait que s_j (ie une règle notée R) $\in$ Res(BC, Prem-Cons, lit_m)

avec lit_m = m ou lit_m = $\neg$m.

Donc R est de la forme:

R: Prem(R2) $\cup$ (Prem(R1) — {lit_m}) $\rightarrow$ Cons(R1)

avec R2 $\in$ BC(Cons, lit_m) et R1 $\in$ BC(Prem, lit_m)}

et s_i = Prem(R) est démontrable dans SF_{BC}

donc Prem(R2) est démontrable dans SF_{BC}

donc Cons(R2) = l est démontrable dans SF_{BC} (en appliquant (E$\rightarrow$)).

Or Prem(R1) — {l} est démontrable dans SF_{BC} puisque Prem(R) est démontrable dans SF_{BC}

donc Prem(R1) est démontrable dans SF_{BC} (en appliquant (I$\wedge$))

donc Cons(R1) = Cons(R) = s_{j+1} est démontrable dans SF_{BC} (par (E$\rightarrow$))

COROLLAIRE du LEMME 7: Soit BC les axiomes d'un SF du type décrit au début du 3) et m un élément de MD_{BC}:

Si ETAUT(BC) = $\emptyset$, BC(Contrad) = $\emptyset$, E $\cap$ MD_{BC} = $\emptyset$ et m $\in$ MD_{BC} alors

SF_{BC} cohérent par rapport à E $\Leftrightarrow$ $SF_{FT(BC,m)}$ cohérent par rapport à E

Démonstration évidente d'après le lemme 6.

Remarque: si E $\cap$ MD_{BC} = $\emptyset$, alors E $\cap$ $MD_{FT(BC,m)}$ = $\emptyset$

b.3) Réunion des couples de règles contradictoires

On va maintenant formaliser l'étape 2 de l'opération de coupure. Pour cela, on définit:

Res(BC,Cons-Cons,"m") = {r: Prem(r') $\cup$ Prem(r") $\rightarrow \perp$,

avec r' $\in$ BC(Cons, Abs(m)) et r" $\in$ BC(Prem, $\neg$Abs(m))}

(rappelons que BC(Cons, m) = {r $\in$ BC / m $\in$ Cons(r)}

et BC(Cons,"m") = BC(Cons,m) $\cup$ BC(Cons,$\neg$m))

RRC^{21}(BC,m) = (BC — BC(Cons,"m")) $\cup$ Res(BC,Cons-Cons,"m")

ie la base de règles initiale dont on a supprimé les règles concluant sur m et où l'on a réuni les couples de règles concluant l'une sur m et l'autre sur $\neg$m comme indiqué à l'étape 2 de l'algorithme du 1)b).

---

[21] RRC(BC,m) signifie:"Réunion des règles de BC contradictoires sur m "

Exemple: $BC = \{ A1 \wedge A2 \rightarrow B \quad A3 \rightarrow C \quad C \wedge A5 \rightarrow \neg B$
$\neg C \wedge \neg A6 \rightarrow B \quad A6 \wedge B \rightarrow E \}$

$RRC(BC,B) = \{A3 \rightarrow C \quad A6 \wedge B \rightarrow E$
$A1 \wedge A2 \wedge C \wedge A5 \rightarrow \perp \quad C \wedge A5 \wedge \neg C \wedge \neg A6 \rightarrow \perp \}$

LEMME 8.1: Soit BC les axiomes d'un SF du type décrit au début du 3) tel que $BC(Prem,"m") = \emptyset$ et $m \in MD_{BC}$, alors

$\forall F$ tq $Abs(F) \cap MD_{BC} = \emptyset$, $\forall l$ tq $Abs(l) \in MO_{BC}$ et $Abs(l) \neq m$

$F \vdash_{BC} l \quad \Leftrightarrow \quad F \vdash_{BC - BC(Cons,"m")} l$

Démonstration

$\Leftarrow$: Evident

$\Rightarrow$:

On démontre la contraposée: $non(F \vdash_{BC - BC(Cons,"m")} l) \Rightarrow non(F \vdash_{BC} l)$

Puisque $BC(Prem,"m") = \emptyset$ alors $(BC - BC(Cons,"m"))(Prem,"m") = \emptyset$
Donc m est une hypothèse "morte", ie présente dans aucune partie gauche d'un axiome du SF, donc le fait de rajouter "m" en hypothèse initiale ne permet pas de déduire quoique ce soit de plus.
si $Abs(l) <> m$ et $non(F \vdash_{BC - BC(Cons,"m")} l)$
alors $non(F \cup \{lit_m\} \vdash_{BC - BC(Cons,"m")} l))$ avec $lit_m = m$ ou $lit_m = \neg m$
Or si $(F \vdash_{BC(Cons,"m")} p)$ alors $Abs(p) = m$,
donc si $non(F \vdash_{BC - BC(Cons,"m")} l)$ alors $non(F \vdash_{BC} l)$ ou $Abs(l) = m$

LEMME 8.2: Soit BC les axiomes d'un SF du type décrit au début du 3) tel que $BC(Prem,"m") = \emptyset$ et $m \in MD_{BC}$, alors $\forall F$ tq $Abs(F) \cap MD_{BC} = \emptyset$,

$F \vdash_{BC} m \wedge \neg m$ ou $F \vdash_{BC} \perp \quad \Leftrightarrow \quad F \vdash_{RRC(BC,m)} \perp$

Démonstration

$\Rightarrow$:

-> Si $F \vdash_{BC} m \wedge \neg m$ alors $F \vdash_{BC} m$ et $F \vdash_{BC} \neg m$

Or $m \in MD_{BC}$ donc $\exists$**r1** $\in$ **BC(Cons, m)**, $\exists$**r2** $\in$ **BC(Cons, $\neg$m)** tel que

$F \vdash_{BC} Prem(r1)$ et $F \vdash_{BC} Prem(r2)$ (m et $\neg m$ doivent être inférés)

donc $\forall l \in Prem(r1) \cup Prem(r2)$, $F \vdash_{BC} l$ et $Abs(l) \neq m$ puisque $BC(Prem,"m") = \emptyset$.

Donc, puisque $Abs(F) \cap MD_{BC} = \emptyset$, on sait (Lemme 8.1) que:

$\forall l \in Prem(r1) \cup Prem(r2)$, $F \vdash_{BC - BC(Cons,"m")} l$

donc $F \vdash_{RRC(BC,m)} l$ donc **$F \vdash_{RRC(BC,m)} Prem(r1) \wedge Prem(r2)$** par applications successives de ($I\wedge$).

Or, on sait que si $r1 \in BC(Cons, m)$ et $r2 \in BC(Cons, \neg m)$,

$\exists R \in RRC(BC,m)$ tq $Prem(R) = Prem(r1) \cup Prem(r2)$, et $Cons(R) = \perp$

donc F $\vdash_{RRC(BC,m)}$ Prem(R) et F $\vdash_{RRC(BC,m)} \perp$ par application de (E→).
-> Enfin F $\vdash_{BC} \perp \Rightarrow$ (lemme 8.1) F $\vdash_{BC - BC(Cons,"m")} \perp \Rightarrow$ F $\vdash_{RRC(BC,m)} \perp$

$\Leftarrow$: si F $\vdash_{RRC(BC,m)} \perp$ alors $\exists r \in$ **RRC(BC,m)** tq **Cons(r)** = $\perp$
et F $\vdash_{RRC(BC,m)}$ **Prem(r)**
Or F $\vdash_{RRC(BC,m)}$ Prem(r) $\Rightarrow$ F $\vdash_{RRC(BC,m) - BC(\perp)}$ Prem(r)
puisque toutes les règles de BC($\perp$) déduisent $\perp$ que l'on ne rencontre jamais en prémisse de règle. Donc F $\vdash_{BC - BC(Cons,"m")}$ Prem(r), donc F $\vdash_{BC}$ Prem(r).
De plus
* soit r $\notin$ Res(BC,Cons-Cons,"m") $\Rightarrow$ r $\in$ BC $\Rightarrow$ F $\vdash_{BC} \perp$ ((E→))
* soit r $\in$ Res(BC,Cons-Cons,"m"). Dans ce cas, il existe r1 $\in$ BC(Cons,m)
et r2 $\in$ BC(Cons,¬m) tq Prem(r1) $\cup$ Prem(r2) = Prem(r) donc
puisque F $\vdash_{BC}$ Prem(r), nécessairement F $\vdash_{BC}$ Prem(r1) et F $\vdash_{BC}$ Prem(r2)
donc F $\vdash_{BC}$ m et F $\vdash_{BC}$ ¬m (application de (E→))
donc F $\vdash_{BC}$ m $\wedge$ ¬m (application de (I$\wedge$))

LEMME 9: Soit BC les axiomes d'un SF du type décrit au début du 3)
tel que BC(Prem,"m") = $\emptyset$ et E tq E $\cap$ MD_{BC} = $\emptyset$, si m $\in$ MD_{BC},
SF_{BC} **cohérent** par rapport à E $\Leftrightarrow$ $SF_{RRC(BC,m)}$ **cohérent** par rapport à E (et E $\cap$ $MD_{RRC(BC,m)}$ = $\emptyset$)

Démonstration
SF_{BC} cohérent par rapport à E et E $\cap$ MD_{BC} = $\emptyset$
$\Leftrightarrow \forall$F tq $Valid_E$(F) et $\forall$p, non(F $\vdash_{BC}$ p $\wedge$ ¬p) et non(F $\vdash_{BC} \perp$) .
$\Leftrightarrow \forall$F tq $Valid_E$(F), d'une part: $\forall$p tq p $\neq$ m , non(F $\vdash_{BC}$ p $\wedge$ ¬p)
et d'autre part: non(F $\vdash_{BC}$ m $\wedge$ ¬m) et non(F $\vdash_{BC} \perp$) .
$\Leftrightarrow \forall$F tq $Valid_E$(F),
d'une part: $\forall$p tq p $\neq$ m, non(F $\vdash_{RRC(BC,m)}$ p $\wedge$ ¬p) (lemme 8.1)
d'autre part: non(F $\vdash_{RRC(BC,m)} \perp$) (lemme 8.2)
$\Leftrightarrow$ $SF_{RRC(BC,m)}$ cohérent par rapport à E

Remarque: $MD_{RRC(BC,m)}$ = MD_{BC} — {m}
donc E $\cap$ MD_{BC} = $\emptyset \Rightarrow$ E $\cap$ $MD_{RRC(BC,m)}$ = $\emptyset$

b.4) Coupure sur un mot déductible

Soit SF_{BC} un système formel du type décrit au début du 3) et d'axiomes BC, et m un élément de MD_{BC}, on définit l'opération de coupure sur m dans BC par:
$COUP^{22}(BC,m) = RRC(FT(BC, m), m)$

Exemple: $BC = \{A1 \wedge A2 \rightarrow B \quad A2 \rightarrow E \quad C \wedge A5 \rightarrow \neg B$
$\neg C \wedge \neg A6 \rightarrow B \quad A7 \wedge B \rightarrow E \quad A8 \rightarrow \neg E\}$

$COUP(BC,B) = \{A2 \rightarrow E \quad A8 \rightarrow \neg E$
$A7 \wedge A1 \wedge A2 \rightarrow E \quad A7 \wedge \neg C \wedge \neg A6 \rightarrow E$
$A1 \wedge A2 \wedge C \wedge A5 \rightarrow \perp \quad \neg C \wedge \neg A6 \wedge C \wedge A5 \rightarrow \perp\}$

LEMME 10: Si **$STAUT(SRIC(BC)) = BC$**, E tq $E \cap MD_{BC} = \emptyset$ et $m \in MD_{BC}$ **alors**
SF_{BC} cohérent par rapport à E $\Leftrightarrow$ **$SF_{COUP(BC,m)}$ cohérent** par rapport à E (et $E \cap MD_{COUP(BC,m)} = \emptyset$)

Démonstration
SF_{BC} cohérent par rapport à E et $E \cap MD_{BC} = \emptyset$, $STAUT(BC) = BC$,
$BC(Contrad) = \emptyset$ et $m \in MD_{BC}$

$\Leftrightarrow$ $SF_{FT(BC,\ m)}$ est cohérent par rapport à E (corollaire du lemme 7),
et $FT(BC,m)(Prem,"m") = \emptyset$, $MD_{BC} \supset MD_{FT(BC,m)}$ donc $E \cap MD_{FT(BC,m)} = \emptyset$

$\Leftrightarrow$ $SF_{RRC(FT(BC,\ m),\ m)} = SF_{COUP(BC,m)}$ est cohérent par rapport à E (Lemme 9)
et $E \cap MD_{COUP(BC,m)} = \emptyset$ (d'après la remarque du lemme 9) .

Remarque: **La base de règles, après coupure sur le littéral m, ne contient plus m ou $\neg$m: $(COUP(BC,\ m))("m") = \emptyset$**
En effet, l'opération de fermeture transitive supprime les occurrences de m en prémisse de règle, et celle de réunion des règles contradictoires enleve les occurrences en conséquent de règle.

[22] COUP(BC, m) signifie "Coupure dans BC sur le mot m"

c) Algorithme général

Les différentes transformations décrites jusqu'ici vont s'enchaîner comme suit:

Algorithme de recherche de la BC-Cohérence de SF_{b0}

b1 = SSUB(STAUT(SRIC(b0)))
b2 = SRINU(b1, MD_{b0})
w0 = b2, m = Premier_Elément_De(MD_{b0})
TANT_QUE m ≠ NIL et w0 ≠ Ø
SI m ∈ MD_{w0}
ALORS
w1 = FT(w0, m)
w2 = SRINU*(SSUB(STAUT(SRIC(w1))), MD_{b0})
w3 = RRC(w2, m)
w0 = SRINU*(SSUB(STAUT(w3)), MD_{b0})
m = Elément_Suivant_De(MD_{b0})
FIN_TANT_QUE
bfin = w0

REMARQUES:
1°) SRINU* correspond à l'opération SRINU décrite au 3)a.3.3 (construction du sous-ens. max. utile pour la cohérence). Mais ici, cette opération n'est activée que si l'on a détecté, dans les phases antérieures de l'algorithme, qu'elle était nécessaire: par exemple, si lors de l'élimination de la tautologie $A \wedge B \rightarrow A$, on s'aperçoit qu'on vient de supprimer la dernière occurrence de A en conséquent (POSC[A] = 0), et que cette suppression va permettre d'éliminer de nouvelles règles du système (NEGC[A] ≠ 0 ou POSP[A] ≠ 0), on activera la recherche de SRINU sur la base courante et on mettra A en tête de la liste des entités sur lesquelles s'effectue la recherche.

2°) La séquence d'instruction qui constitue le corps du ALORS correspond à une opération de coupure sur un littéral de MD_{b0}, avec élimination simultanée de toutes les règles "parasites", au sens de la cohérence, générées par l'opération.
En effet, COUP(BC, m) est défini par: COUP(BC,m) = RRC(FT(BC, m), m), c'est-à-dire par l'enchaînement des deux opérations FT et RRC. Ici, on "purge" le système de ces règles inutiles, après chacune de ces transformations.

Exemple: Si l'on reprend l'exemple du b.4), on a vu que

BC = {$A1 \wedge A2 \rightarrow B$ $A2 \rightarrow E$ $C \wedge A5 \rightarrow \neg B$
$\neg C \wedge \neg A6 \rightarrow B$ $A7 \wedge B \rightarrow E$ $A8 \rightarrow \neg E$}

COUP(BC,B) = {$A2 \rightarrow E$ $A8 \rightarrow \neg E$
$A7 \wedge A1 \wedge A2 \rightarrow E$ $A7 \wedge \neg C \wedge \neg A6 \rightarrow E$
$A1 \wedge A2 \wedge C \wedge A5 \rightarrow \perp$ $\neg C \wedge \neg A6 \wedge C \wedge A5 \rightarrow \perp$ }

Appelons COUP_ELIM(BC, m) l'opération constituant le corps du ALORS on a ici:

COUP_ELIM(BC,B) = $\{A2 \rightarrow E \quad A8 \rightarrow \neg E$
$A7 \wedge \neg C \wedge \neg A6 \rightarrow E \quad A1 \wedge A2 \wedge C \wedge A5 \rightarrow \perp\}$

car SSUB après FT éliminera $A7 \wedge A1 \wedge A2 \rightarrow E$ qui est subsumée par $A2 \rightarrow E$, et STAUT, après RRC, élimine la règle $\neg C \wedge \neg A6 \wedge C \wedge A5 \rightarrow \perp$ qui est intrinsèquement illogique.

Ce que nous avions remarqué après le lemme 10 à propos de COUP reste bien sur ici valable: chaque opération de COUP_ELIM correspond à la suppression dans le SF courant de toutes les occurrences d'un élément de MD_{b0}.

THEOREME 2:
SF_{b0} BC-cohérent $\Leftrightarrow$ SF_{bfin} cohérent par rapport à MNR_{b0}

Démonstration
On a vu que SF_{b0} BC-cohérent ie SF_{b0} cohérent par rapport à MNR_{b0}
$\Leftrightarrow$ $SF_{SRIC(b0)}$ cohérent par rapport à MNR_{b0}
(lemme 6)
$\Leftrightarrow$ $SF_{STAUT(SRIC(b0))}$ cohérent par rapport à MNR_{b0}
(lemme 1)
$\Leftrightarrow$ $SF_{b1} = SF_{SSUB(STAUT(SRIC(b0)))}$ cohérent par rapport à MNR_{b0}
(lemme 3)
$\Leftrightarrow$ $SF_{b2} = SF_{SRINU(b1, MDb0)}$ cohérent par rapport à MNR_{b0}
puisque $MNR_{b0} \cap MD_{b1} = \emptyset$ et $MNR_{b0} \cap MD_{b0} = \emptyset$ (Théorème 1)

Et, à l'intérieur du TANT_QUE, on sait que:
- $\forall i$, $MNR_{b0} \cap MD_{wi} = \emptyset$, (remarque du lemme 9),
- ETAUT(w0) = $\emptyset$ et wo(Contrad) = $\emptyset$

donc:
SF_{w0} cohérent par rapport à MNR_{b0}
$\Leftrightarrow$ $SF_{w1} = SF_{FT(w0, m)}$ avec $m \in MD_{w0}$ est cohérent par rapport à MNR_{b0}
(Corollaire du lemme 7)
$\Leftrightarrow$ $SF_{SSUB(STAUT(SRIC(w1)))}$ cohérent par rapport à MNR_{b0}
(lemmes 6, 1 et 3)
$\Leftrightarrow$ SFw2= SFSRINU(SSUB(STAUT(SRIC(w1))), MD_{b0}) cohérent par rapport à MNRb0
puisque $MNR_{b0} \cap MD_{SSUB(STAUT(SRIC(w1)))} = \emptyset$ et $MNR_{b0} \cap MD_{b0} = \emptyset$
(Théorème 1)
$\Leftrightarrow$ $SF_{w3} = SF_{RRC(w2, m)}$ cohérent par rapport à MNR_{b0}
puisque w2(Prem,"m") = $\emptyset$ et $m \in MD_{w0}$ donc $m \in MD_{w2}$
(lemme 9)

$\Leftrightarrow SF_{w0} = SF_{SRINU^*(SSUB(STAUT(w3)),\ MDb0)}$ cohérent par rapport à MNR_{b0}
puisque $MNR_{b0} \cap MD_{SSUB(STAUT(w3))} = \emptyset$ et $MNR_{b0} \cap MD_{b0} = \emptyset$
(Théorème 1)

THEOREME 3:
a) Si $VALID_{MNR_{b0}}(F)$,
$(F \vdash_{b0} \bot$ ou $F \vdash_{b0} p \wedge \neg p) \quad \Leftrightarrow \quad F \vdash_{bfin} \bot$
b) La partie prémisse de chaque élément de bfin constitue un ensemble d'hypothèses initiales valides permettant de déduire dans SF_{b0} soit $\bot$, soit un mot et son opposé.
c) tout ensemble d'hypothèses initiales valides permettant de déduire dans SF_{b0} soit $\bot$, soit un mot et son opposé, contient la partie prémisse d'une règle de bfin.
d) $SF_{bfin} = \emptyset \quad \Leftrightarrow$ b0 est BC-cohérente.

Démonstration
a) on a vu que si SRINU(b0) est la partie de b0 permettant de faire des tentatives de démonstrations contradictoires à partir d'hypothèses initiales valides, donc
si $VALID_{MNR_{b0}}(F)$
$F \vdash_{b0} \bot$ ou $F \vdash_{b0} p \wedge \neg p \Leftrightarrow (F \vdash_{SRINU(b0)} \bot$ ou $F \vdash_{SRINU(b0)} p \wedge \neg p)$

Ensuite, on sait que pour chaque opération de coupure sur un littéral m, COUP(BC,m) = RRC(FT(BC, m), m) et :
$F \vdash_{BC} l \Leftrightarrow F \vdash_{FT(BC,m)} l$ (lemme 7) et

- si l tq Abs(l) ≠ m, alors

$F \vdash_{FT(BC,m)} l \Leftrightarrow F \vdash_{FT(BC,m) - FT(BC,m)(Cons, "m")} l$ (lemme 8.1)
$\Leftrightarrow F \vdash_{RRC(FT(BC,m), m)} l$
puisque RRC(X,m) = X — X(Cons,"m")) ∪ Res(X,Cons-Cons,"m") (voir b.3)

- si Abs(l) = m

$F \vdash_{FT(BC,m)} m \wedge \neg m$ ou $F \vdash_{FT(BC,m)} \bot \Leftrightarrow$
$F \vdash_{RRC(FT(BC,m), m)} \bot$ (lemme 8.2)
Donc $F \vdash_{BC} \bot$ ou $F \vdash_{BC} p \wedge \neg p$
$\Leftrightarrow F \vdash_{COUP(BC,m)} \bot$ ou $F \vdash_{COUP(BC,m)} p \wedge \neg p$ si Abs(p) ≠ m,
ou $F \vdash_{COUP(BC,m)} \bot$ sinon.
Et cela pour chaque opération de coupure,
Donc $F \vdash_{SRINU(b0)} \bot$ ou $F \vdash_{SRINU(b0)} p \wedge \neg p$
$\Leftrightarrow F \vdash_{bfin} \bot$

b) Par construction, les règles de bfin :

1°) concluent toutes sur $\perp$, puisque on effectue l'opération de coupure sur tous les littéraux déductibles restants à l'issue de la phase de suppression des règles inutiles et que cette opération remplace toutes les règles concluant sur ce littéral ou sur son opposé par d'autres règles concluant sur $\perp$.

2°) ne contiennent en partie prémisse que des éléments de MNR_{b0}, puisque issues d'un algorithme supprimant toutes les occurrences de tous les éléments de MD_{b0}.

Elles sont donc de la forme: $\wedge l_i \rightarrow \perp$

$Abs(l_i) \in MNR_{b0}$

3°) sont non intrinsèquement illogiques, non redondantes et atteignables puisqu'elles ont passé les filtres SRIC, STAUT, SSUB, SRINU et SRINU*.

Chaque partie prémisse constitue donc une conjonction d'hypothèses initiales valides permettant de déduire l'absurde dans SF_{bfin}.

Donc $\forall r \in$ bfin, $VALID_{MNRBC}(Prem(r))$ et $Prem(r) \vdash_{bfin} \perp$

donc d'après le a)

$VALID_{MNRBC}(Prem(r))$ et $Prem(r) \vdash_{b0} \perp$ ou $Prem(r) \vdash_{b0} p \wedge \neg p$

c) Réciproquement, on sait que, si

$VALID_{MNRBC}(F)$ et $F \vdash_{b0} \perp$ ou $F \vdash_{b0} p \wedge \neg p$

alors $F \vdash_{bfin} \perp$

Or, si F est non-contradictoire, il ne contient pas $\perp$,

donc si F permet de démontrer $\perp$ dans bfin, c'est que $\perp$ est inféré, donc c'est que $\exists r \in$ bfin tel que $F \vdash_{bfin} Prem(r)$.

Or $MNR_{b0} \supset Prem(r)$ et bfin ne permet que d'inférer $\perp$,

donc $F \supset Prem(r)$.

d)

$\Rightarrow$: le SF basé sur la base de règles vide est cohérent par rapport à MNR_{b0} puisqu'il ne permet de construire aucune démonstration, donc en particulier il ne permet pas de déduire ni un mot et son opposé, ni $\perp$.

Donc si bfin = $\emptyset$, bfin est cohérent par rapport à MNR_{b0}

donc, grâce au théorème 2, b0 est BC-Cohérente.

$\Leftarrow$: si b0 BC-cohérent, alors bfin cohérent par rapport à MNR_{b0},

ce qui oblige, d'après le a), que bfin soit vide.

d) Exemple Récapitulatif

On reprend ici la base de règles qui a servi de support à l'exemple de la partie 1) de présentation du traitement. On respectera bien sur ici les étapes successives précises que nous venons de donner dans la description de l'algorithme, et utiliserons les notations définies.

BC = b0 = {
reg1: $A1 \rightarrow B$,
reg2: $\neg B \rightarrow C$,
reg3: $C \rightarrow D$,
reg4: $D \rightarrow E$,
reg5: $F \rightarrow \neg D$,
reg6: $A2 \wedge A3 \rightarrow G$
reg7: $A3 \wedge A8 \rightarrow H$
reg8: $A4 \wedge A6 \rightarrow \neg G$
reg9: $\neg A4 \wedge \neg A7 \rightarrow G$
reg10: $A8 \wedge G \rightarrow H$
reg11: $A9 \wedge \neg A8 \rightarrow \neg H$
reg12: $\neg A4 \wedge \neg A7 \wedge G \rightarrow G$
reg13: $A1 \wedge \neg A3 \rightarrow B$
reg14: $A9 \wedge \neg A9 \rightarrow \perp$ }

et MD_{BC} = MD0 = {B, C, D, E, G, H} . Soit (B, C, D, E, G, H) la liste correspondante

1°) Elimination des règles inutiles

b0' = SRIC(b0) = b0 - {reg14: $A9 \wedge \neg A9 \rightarrow \perp$}

b0'' = STAUT(b0') = b0' - {reg12: $\neg A4 \wedge \neg A7 \wedge G \rightarrow G$}

b1 = SSUB(b0'') = b0'' - {reg13: $A1 \wedge \neg A3 \rightarrow B$}

donc b1 = {
reg1: $A1 \rightarrow B$,
reg2: $\neg B \rightarrow C$,
reg3: $C \rightarrow D$,
reg4: $D \rightarrow E$,
reg5: $F \rightarrow \neg D$,
reg6: $A2 \wedge A3 \rightarrow G$
reg7: $A3 \wedge A8 \rightarrow H$
reg8: $A4 \wedge A6 \rightarrow \neg G$
reg9: $\neg A4 \wedge \neg A7 \rightarrow G$
reg10: $A8 \wedge G \rightarrow H$
reg11: $A9 \wedge \neg A8 \rightarrow \neg H$ }

- ERINU(b1, B) = {reg1: A1 → B} ∪ {reg2: ¬B → C}

donc b1' = {
reg3: C → D,
reg4: D → E,
reg5: F → ¬D,
reg6: A2 ∧ A3 → G
reg7: A3 ∧ A8 → H
reg8: A4 ∧ A6 → ¬G
reg9: ¬A4 ∧ ¬A7 → G
reg10: A8 ∧ G → H
reg11: A9 ∧ ¬A8 → ¬H }

- ERINU(b1', B) = ∅, ERINU(b1', C) = {reg3: C → D}

donc b1" = {
reg4: D → E,
reg5: F → ¬D,
reg6: A2 ∧ A3 → G
reg7: A3 ∧ A8 → H
reg8: A4 ∧ A6 → ¬G
reg9: ¬A4 ∧ ¬A7 → G
reg10: A8 ∧ G → H
reg11: A9 ∧ ¬A8 → ¬H }

- ERINU(b1", B) = ERINU(b1", C) = ∅,
  ERINU(b1", D) = {reg4: D → E, reg5: F → ¬D}

donc b1"' = {
reg6: A2 ∧ A3 → G
reg7: A3 ∧ A8 → H
reg8: A4 ∧ A6 → ¬G
reg9: ¬A4 ∧ ¬A7 → G
reg10: A8 ∧ G → H
reg11: A9 ∧ ¬A8 → ¬H }

- ERINU(b1"', B) = ERINU(b1"', C) = ERINU(b1"', D) = ERINU(b1"', E) = ERINU(b1"', G) = ERINU(b1"', H) = ∅,
  donc b2 = b1"'

2°) Coupure
w0 = b2
-> m = B , $m \notin MD_{w0}$

-> m = C, $m \notin MD_{w0}$

-> m = D, $m \notin MD_{w0}$

-> m = E, $m \notin MD_{w0}$

-> m = G, $\mathbf{m \in MD_{w0}}$

w1 = FT(wo, G)

donc w1 = {
reg6: $A2 \wedge A3 \rightarrow G$
reg7: $A3 \wedge A8 \rightarrow H$
reg8: $A4 \wedge A6 \rightarrow \neg G$
reg9: $\neg A4 \wedge \neg A7 \rightarrow G$
reg11: $A9 \wedge \neg A8 \rightarrow \neg H$
regFT(10,6): $A8 \wedge A2 \wedge A3 \rightarrow H$
regFT(10,9): $A8 \wedge \neg A4 \wedge \neg A7 \rightarrow H$ }

SRIC(w1) = w1
STAUT(w1) = w1
SSUB(w1) = w1 - {regFT(10,6): $A8 \wedge A2 \wedge A3 \rightarrow H$} (la règle est subsumée par $A3 \wedge A8 \rightarrow H$)
w2 = SRINU*(SSUB(w1), MD_{b0}) = SSUB(w1) = w1 - {regFT(10,6): $A8 \wedge A2 \wedge A3 \rightarrow H$}

w3 = RRC(w2, G)

donc w3 = {
reg7: $A3 \wedge A8 \rightarrow H$
reg11: $A9 \wedge \neg A8 \rightarrow \neg H$
regFT(10,9): $A8 \wedge \neg A4 \wedge \neg A7 \rightarrow H$
reg CONTRAD(6,8) : $A2 \wedge A3 \wedge A4 \wedge A6 \rightarrow \perp$
reg CONTRAD(9,8) : $\neg A4 \wedge \neg A7 \wedge A4 \wedge A6 \rightarrow \perp$ }

w3' = STAUT(w3) = w3 - {reg CONTRAD(9,8) : $\neg A4 \wedge \neg A7 \wedge A4 \wedge A6 \rightarrow \perp$}

w0 = SRINU*(SSUB(w3'), MD_{b0}) = w3'

donc w0 = {
reg7: $A3 \wedge A8 \rightarrow H$
reg11: $A9 \wedge \neg A8 \rightarrow \neg H$
regFT(10,9): $A8 \wedge \neg A4 \wedge \neg A7 \rightarrow H$
reg CONTRAD(6,8) : $A2 \wedge A3 \wedge A4 \wedge A6 \rightarrow \perp$ }

-> m = H, $\mathbf{m \in MD_{w0}}$
w1 = FT(w0, H) = w0
donc w2 = SRINU*(SSUB(STAUT(SRIC(w1))), MD_{b0}) = w0

w3 = RRC(w2, H)

donc w3 = {
reg CONTRAD(6,8) : $A2 \wedge A3 \wedge A4 \wedge A6 \rightarrow \perp$
regCONTRAD(7, 11) : $A3 \wedge A8 \wedge A9 \wedge \neg A8 \rightarrow \perp$
regCONTRAD(FT(10,9), 11) : $A8 \wedge \neg A4 \wedge \neg A7 \wedge A9 \wedge \neg A8 \rightarrow \perp$ }

w3' = STAUT(w3) =
w3 - {regCONTRAD(7, 11) : $A3 \wedge A8 \wedge A9 \wedge \neg A8 \rightarrow \perp$
regCONTRAD(FT(10,9), 11) : $A8 \wedge \neg A4 \wedge \neg A7 \wedge A9 \wedge \neg A8 \rightarrow \perp$}

w0 = SRINU*(SSUB(w3'), MD_{b0}) = w3 = {reg CONTRAD(6,8) : $A2 \wedge A3 \wedge A4 \wedge A6 \rightarrow \perp$}

m = NIL

bfin = w0 = {reg CONTRAD(6,8) : $A2 \wedge A3 \wedge A4 \wedge A6 \rightarrow \perp$}

3) Approche sémantique de l'algorithme de vérification de la BC-Cohérence de MELOMIDIA

a) Introduction

On a vu au 4)b) du chapitre précédent que la BC-Cohérence pouvait se traduire en terme de contraintes sur la réalisation de la théorie concernée. Nous allons reprendre ici les lemmes de la partie précédente, et en faire la démonstration par la théorie des modèles.

Nous verrons que dans ce nouveau formalisme, les démonstrations sont beaucoup plus simples, ce qui permet d'économiser certaines étapes nécessaires dans les démonstrations syntaxiques. On ne s'étonnera donc pas de ne trouver ici que les démonstrations des lemmes principaux de la partie précédente. Par ailleurs, certains points de détail ne seront pas retraités ici, étant donnée l'existence des démonstrations exactes dans la partie précédente.

Enfin, l'objectif de cette partie est aussi de dessiner un cadre de démonstration des algorithmes du même genre que celui de MELOMIDIA. Nous espérons donner l'exemple en proposant la démonstration de la validité de notre algorithme dans ce nouveau cadre. Rappelons que le travail de la preuve de la correction d'un algorithme est le seul qui permette vraiment de cerner les frontières d'application de cet algorithme. Que les auteurs des travaux apparentés s'inspirent de ce travail pour construire une "vraie" preuve de leurs traitements nous comblerait pleinement.

b) Rappels

SF_{BC} est cohérente par rapport à E si $\forall BF$ tq $VALID_E(BF)$, alors il existe

- $i0_{BF}$: Abs(BF) --> {V, F} telle que

$i0_{BF}(x) = V$ si $x \in BF$ et $i0_{BF}(x) = F$ si $\neg x \in BF$

- $i1_{BF}$: MO_{BC} - Abs(BF) --> {V, F, I} telle que

si on note i la réunion de $i0_{BF}$ et $i1_{BF}$, on a : $\forall R \in BC$, $vv_i(R) = V$

On rappelle la table des opérateurs $\wedge$ et $\rightarrow$:

$\wedge$	V	F	I
V	V	F	I
F	F	F	F
I	I	F	I

$\rightarrow$	V	F	I
V	V	F	F
F	V	V	V
I	V	V	V

c) Démonstration du Lemme 1

Rappel du Lemme : (cf 2)a)a.1)

SF_{BC} cohérente par rapport à E ($E \cap MD_{BC} = \emptyset$) $\Leftrightarrow$ **$SF_{STAUT(BC)}$ cohérente** par rapport à E

On sait que le sens $\Rightarrow$ est trivial.

$\Leftarrow$: Supposons que BC' = STAUT(BC) soit cohérent par rapport à E

avec BC' = BC – (ETAUT1(BC) $\cup$ ETAUT2(BC))

et ETAUT1(BC) = {R $\in$ BC / $\exists l \in MO_{BC}$ tq $l \in$ Prem(R) et $\neg l \in$ Prem(R)}

ETAUT2(BC) = {R $\in$ BC / $\exists l \in MO_{BC}$ tq $l \in$ Prem(R) et $l \in$ Cons(R)}

Les règles de ETAUT1(BC) sont de la forme

R: $p \wedge \neg p \wedge Q \rightarrow r$ avec p et r appartenant à MO_{BC} et $\mathcal{P}(MO_{BC}) \supset Q$

On prolonge i, réalisation de BC' dans {V, F, I} sur MO_{BC} - ($MO_{BC'} \cup E$) de la façon suivante: $\forall x \in MO_{BC}$ - ($MO_{BC'} \cup E$) , i(x) = V.

Dressons la table de vérité de R. On a:

p	V	F	I
$\neg p$	F	V	I
$p \wedge \neg p$	F	F	I
$p \wedge \neg p \wedge Q$	F	F	I
R: $p \wedge \neg p \wedge Q \rightarrow r$	V	V	V

De même, les règles de ETAUT2(BC) sont de la forme:

R: $p \wedge Q \rightarrow p$ avec p appartenant à MO_{BC} et $\mathcal{P}(MO_{BC}) \supset Q$

Dressons la table de vérité de R. On a:

p	V	F	I
$p \wedge Q$	$vv_i(Q)$	F	F / I[23]
R: $p \wedge Q \rightarrow p$	V	V	V

Donc toutes les règles de BC' ont la valeur de vérité VRAIE dans cette interprétation. Donc tout modèle de BC' se prolonge de manière évidente en modèle de BC.

[23] "F/I" signifie "F ou I". Plus généralement, on écrira "X /Y" plutôt que "X ou Y"

d) Démonstration du Lemme 2

Il s'agit de montrer (cf 2)a)a.2) que si l'on a
R: Prem(R) → Cons(R) et
R': Prem(R) ∧ Q → Cons(R') avec $\mathcal{P}(MO_{BC}) \supset Q$
Tout modèle de BC' = BC - {R'} est un modèle de BC dans les trois cas qui vont suivre:

On reprend les mêmes notations et on étend i comme ci-dessus.

1er Cas : Cons(R) = Cons(R')
dans le modèle de BC', on a $vv_i(R) = V$, et

- soit $vv_i(Cons(R)) = V$, donc $vv_i(Cons(R')) = V$ donc $vv_i(R') = V$
- soit $vv_i(Cons(R)) = F/I \Rightarrow vv_i(Prem(R)) = F/I \Rightarrow vv_i(Prem(R) \wedge Q) = F/I$
  $\Rightarrow vv_i(R') = V$

2ème Cas : Cons(R) = ⊥
On sait que ∀i $vv_i(\bot) = F$. Et, dans le modèle de BC', on a $vv_i(R) = V$
donc $vv_i(Prem(R)) = F / I \Rightarrow vv_i(Prem(R) \wedge Q) = F/I \Rightarrow vv_i(R') = V$

3ème Cas : R' est de la forme R' : Prem(R) ∧ ¬Cons(R) ∧ Q → Cons(R')
avec $\mathcal{P}(MO_{BC}) \supset Q$.
Ecrivons les valeurs de vérité de cette formule, en fonction des valeurs possibles pour Prem(R) et Cons(R) sachant que $vv_i(R) = V$:

Prem(R)	V	F	I
Cons(R)	V	V / F / I	V / F / I
¬Cons(R)	F	F / V / I	F / V / I
Prem(R) ∧ ¬Cons(R)	F	F	F / I
Prem(R) ∧ ¬Cons(R) ∧ Q	F	F	F / I
Prem(R) ∧ ¬Cons(R) ∧ Q → Cons(R')	V	V	V

donc, la valeur de vérité de R' est VRAIE dans tous les cas.
Donc tout modèle de BC - {R'} s'étend trivialement à BC.

e) Démonstration du Lemme 5

Rappel du Lemme : (cf 2)a)a.3) a.3.3))

Si E est tel que $E \cap MD_B = \emptyset$,

SF_B cohérent par rapport à E $\Leftrightarrow \forall m \in MO_B - E$, $SF_{(B - ERINU(B,m))}$ cohérent par rapport à E

On ne démontre que le sens "$\Leftarrow$". On reprend pour cela le tableau du 2)a)a.3)a.3.2):

POSC(m)	NEGC(m)	POSP(m)	NEGP(m)	ERINU(BC,m)
0	0	0 ou 1	0 ou 1	BC("m")
1	0	0	0 ou 1	BC("m")
1	0	1	0 ou 1	BC(Prem,¬m)
0	1	0 ou 1	0	BC("m")
0	1	0 ou 1	1	BC(Prem,m)
1	1	0 ou 1	0 ou 1	∅

On pose BC' = BC — ERINU(BC, m). Soit i la réalisation sur laquelle est construite le modèle de BC' considéré. On étend i à BC de la manière suivante:

$\forall x \in MO_{BC} - (MO_{BC'} \cup E \cup \{m\})$, i(x) = V. On va maintenant discuter suivant les différents cas de figure mentionnés par le tableau ci-dessus.

cas 1: On sait que **m ∉ BC'** puisque ERINU(BC,m) contient toutes les règles où est mentionné m et que $m \in MO_B - E$. On choisit d'étendre i à m de la manière suivante: **i(m) = I**.

On sait que dans ce cas, $vv_i(m) = vv_i(\neg m) = I$ donc

$\forall Q$ tq $MO_{BC} \supset Q$, $vv_i(m \wedge Q) = vv_i(\neg m \wedge Q) = F / I$ donc

$\forall R$, si $m \in Abs(Prem(R))$, $vv_i(Prem(R)) = F / I$ donc $vv_i(R) = V$

Or ERINU(BC, m) ne contient que des règles ayant m ou ¬m en prémisse,

donc $\forall R \in ERINU(BC, m)$, $vv_i(R) = V$

cas 2: Ici aussi **m ∉ BC'** puisque ERINU(BC,m) contient toutes les règles où est mentionné m et que $m \in MO_B - E$. On choisit d'étendre i de la manière suivante: **i(m) = V**. Dans ce cas, $vv_i(m) = V$ et $vv_i(\neg m) = F$

donc $\forall R$, si Cons(R) = m, $vv_i(R) = V$

et $\forall R$, si $\neg m \in Prem(R)$, $vv_i(Prem(R)) = F$ donc $vv_i(R) = V$

Or ERINU(BC, m) ne contient que des règles ayant m en conséquent ou ¬m en prémisse,

donc $\forall R \in ERINU(BC, m)$, $vv_i(R) = V$

cas 3 et 5: voir plus loin

cas 4: On a encore **m ∉ BC'**. On choisit d'étendre i de la manière suivante:
i(m) = F. Dans ce cas, $vv_i(m) = F$ et $vv_i(\neg m) = V$

donc $\forall R$, si $Cons(R) = \neg m$, $vv_i(R) = V$

et $\forall R$, si $m \in Prem(R)$, $vv_i(Prem(R)) = F \Rightarrow vv_i(R) = V$

Or ERINU(BC, m) ne contient que des règles ayant $\neg m$ en conséquent ou m en prémisse,

donc $\forall R \in$ ERINU(BC, m), $vv_i(R) = V$

cas 3 :

Dans ce cas, **m ∈ BC'**. Plus précisément, on sait que dans BC', m apparait uniquement dans les règles de la forme:

R1: $Q \rightarrow m$ avec $m \notin Abs(Q)$ et $\mathcal{P}(MO_{BC'}) \supset Q$

ou R2: $m \wedge P \rightarrow r$ avec $m \notin Abs(P)$, $\mathcal{P}(MO_{BC'}) \supset P$ et $r \in MO_{BC'}$

et que BC contient en plus de BC' l'ensemble ERINU(BC, m) regroupant les règles de la forme: ER: $\neg m \wedge S \rightarrow t$ avec $m \notin Abs(S)$, $\mathcal{P}(MO_{BC}) \supset S$ et $t \in MO_{BC'}$

Montrons que les valeurs de $vv_i(m)$ imposées par le modèle de BC' permettent d'étendre i pour qu'elle devienne un modèle de BC.

On sait que $\forall R \in$ BC', $vv_i(R) = V$ donc :

- soit $\exists R$, $R \in$ BC' et R: $Prem(R) \rightarrow m$ tel que $vv_i(Prem(R)) = V$.

Dans ce cas, on a forcément $vv_i(m) = V$, et donc

$\forall ER \in$ ERINU(BC, m), $vv_i(Prem(ER)) = F \Rightarrow vv_i(ER) = V$, donc le modèle de BC' est également un modèle de BC.

- sinon $\forall R \in$ BC' , si R de type R1, $vv_i(Prem(R)) = F / I$

$\Rightarrow \forall R \in$ BC' , $\forall$ $vv_i(m)$, si R de type R1 alors $vv_i(R) = V$. Et,

- soit $\exists R$, $R \in$ BC' et R: $m \wedge P \rightarrow Cons(R)$ tel que $vv_i(Cons(R)) = F / I$.

Dans ce cas, on sait qu'on a $vv_i(Prem(R)) = vv_i(m \wedge P) = F / I$ ie
$vv_i(m) = F / I$ si P est tel que $vv_i(P) = V$, ou
$vv_i(m)$ quelconque si $vv_i(P) = F / I$

- sinon $\forall R \in$ BC' , si R de type R2, $vv_i(Cons(R)) = V$.

$\Rightarrow \forall R \in$ BC' , $\forall$ $vv_i(m)$, si R de type R2 alors $vv_i(R) = V$.

Donc le modèle de BC', dans ce cas, impose au plus que $vv_i(m)$ vale FAUX ou, indifféremment, I.

Donc si l'on choisit pour BC l'interprétation i' définie par:
i': MO_{BC} --> {V, F, I} telle que
- $\forall x \in MO_{BC'} \cup E - \{m\}$, $i'(x) = i(x)$
- $i'(m) = I$
- $\forall x \in MO_{BC} - (MO_{BC'} \cup E - \{m\})$, $i'(x) = V$

Alors on a:

- $\forall R \in BC'$, $vv_{i'}(R) = vv_i(R)) = V$
- $\forall ER \in ERINU(BC, m)$, $vv_{i'}(Prem(ER)) = F / I \Rightarrow vv_{i'}(ER) = V$

donc i' est un modèle de BC'.

cas 5 :
Il est presque identique au cas 3 en remplaçant m par $\neg m$:
On a aussi **$m \in BC'$.** Plus précisément, on sait que dans BC', m apparait uniquement dans les règles de la forme:

R1: $Q \rightarrow \neg m$ avec $m \notin Abs(Q)$ et $\mathcal{P}(MO_{BC'}) \supset Q$

ou R2: $\neg m \wedge P \rightarrow r$ avec $m \notin Abs(P)$, $\mathcal{P}(MO_{BC'}) \supset P$ et $r \in MO_{BC'}$
et que BC contient en plus de BC' l'ensemble ERINU(BC, m) regroupant les règles de la forme: ER: $m \wedge S \rightarrow t$ avec $m \notin Abs(S)$, $\mathcal{P}(MO_{BC}) \supset S$ et $t \in MO_{BC}$

Montrons que les valeurs de $vv_i(m)$ imposées par le modèle de BC' permettent d'étendre i pour qu'elle devienne un modèle de BC.

On sait que $\forall R \in BC'$, $vv_i(R) = V$ donc :

- soit $\exists R$, $R \in BC'$ et R: $Prem(R) \rightarrow \neg m$ tel que $vv_i(Prem(R)) = V$.

Dans ce cas, on a forcément $vv_i(m) = F$, et donc
$\forall ER \in ERINU(BC, m)$, $vv_i(Prem(ER)) = F \Rightarrow vv_i(ER) = V$, donc le modèle de BC' est également un modèle de BC.

- sinon $\forall R \in BC'$, si R de type R1, $vv_i(Prem(R)) = F / I$

$\Rightarrow \forall R \in BC'$, $\forall\ vv_i(m)$, si R de type R1 alors $vv_i(R) = V$. Et,

- soit $\exists R$, $R \in BC'$ et R: $\neg m \wedge P \rightarrow Cons(R)$ tel que $vv_i(Cons(R)) = F / I$.

Dans ce cas, on sait qu'on a $vv_i(Prem(R)) = vv_i(\neg m \wedge P) = F / I$ ie
$vv_i(\neg m) = F / I$ ie $vv_i(m) = V / I$ si P est tel que $vv_i(P) = V$, ou
$vv_i(\neg m)$ quelconque si $vv_i(P) = F / I$

- sinon $\forall R \in BC'$, si R de type R2, $vv_i(Cons(R)) = V$.

$\Rightarrow \forall R \in BC'$, $\forall\ vv_i(m)$, si R de type R2 alors $vv_i(R) = V$.

Donc le modèle de BC' impose au plus que $vv_i(m)$ vale VRAI ou, indifféremment, I.

Donc si l'on choisit pour BC l'interprétation i' définie par:
i': MO_{BC} --> {V, F, I} telle que
- $\forall x \in MO_{BC'} \cup E - \{m\}$, $i'(x) = i(x)$
- $i'(m) = I$
- $\forall x \in MO_{BC} - (MO_{BC'} \cup E - \{m\})$, $i'(x) = V$

Alors on a:
- $\forall R \in BC'$, $vv_{i'}(R) = vv_i(R)) = V$
- $\forall ER \in ERINU(BC, m)$, $vv_{i'}(Prem(ER)) = F / I \Rightarrow vv_{i'}(ER) = V$

Donc i' est un modèle de BC'.

Conclusion:
Dans tous les cas, le modèle de BC' s'étend facilement pour devenir un modèle de BC. CQFD .

f) Démonstration du Lemme 6

Rappel du Lemme: (cf 2)b)b.1))
Il s'agit de démontrer que si on remplace la règle
R: $\neg Cons(R) \wedge Q \rightarrow Cons(R)$ avec $\mathcal{P}(MO_{BC}) \supset Q$
par la règle R': $\neg Cons(R) \wedge Q \rightarrow \perp$,
on ne change pas la propriété de BC-Cohérence

Soit BC' = BC - {R} $\cup$ {R'}. Montrons que BC et BC' admettent le même modèle.

$\Rightarrow$: $vv_i(R) = V$

Donc soit $vv_i(Q) = F$ et dans ce cas $vv_i(Prem(R')) = F \Rightarrow vv_i(R') = V$
soit $vv_i(Q) = V$ et dans ce cas $vv_i(Cons(R)) = V / I$

(en effet si $vv_i(Cons(R)) = F$, $vv_i(\neg Cons(R)) = V$ donc $vv_i(Prem(R)) = V$
et $vv_i(Cons(R)) = F$ donc $vv_i(R) = F$)
donc $vv_i(\neg Cons(R)) = F / I$
$\Rightarrow vv_i(Prem(R')) = F / I$
$\Rightarrow vv_i(R') = V$

$\Leftarrow$: $vv_i(R') = V$
ie $vv_i(Prem(R')) = F / I \Rightarrow vv_i(Prem(R)) = F / I$
$\Rightarrow vv_i(R) = V$

g) Démonstration du Lemme 10

Rappel du Lemmè: (cf 2)b)b.4)

Si STAUT(SRIC(BC)) = BC, E tq $E \cap MD_{BC} = \varnothing$ et $m \in MD_{BC}$ alors

(BC cohérente par rapport à E) $\Leftrightarrow$ (COUP(BC,m) cohérente par rapport à E).

L'opération de coupure sur un littéral m consiste à remplacer:

. les quatre ensembles de règles:

E1 = BC(Cons, m) = $\{r_1^j \in$ BC de la forme $r_1^j : A_1^j \rightarrow m\}$

E2 = BC(Cons, $\neg$m) = $\{r_2^j \in$ BC de la forme $r_2^j : A_2^j \rightarrow \neg m\}$

E3 = BC(Prem, m) = $\{r_3^j \in$ BC de la forme $r_3^j : m \wedge A_3^j \rightarrow B_3^j\}$

E4 = BC(Prem, $\neg$m) = $\{r_4^j \in$ BC de la forme $r_4^j : \neg m \wedge A_4^j \rightarrow B_4^j\}$

avec $E_i \cap E_j = \varnothing$ puisque STAUT(SRIC(BC)) = BC

. par les trois ensembles de règles:

F1 = Res(BC, Prem-Cons, m) = $\{ r_{1,3}^{j,k} : A_1^j \wedge A_3^k \rightarrow B_3^k$, avec

$r_1^j \in$ BC de la forme $r_1^j : A_1^j \rightarrow m$ et $r_3^k \in$ BC de la forme $r_3^k : m \wedge A_3^k \rightarrow B_3^k\}$

F2 = Res(BC, Prem-Cons, $\neg$m) =$\{ r_{2,4}^{j,k}: A_2^j \wedge A_4^k \rightarrow B_4^k$, avec

$r_2^j \in$ BC de la forme $r_2^j : A_2^j \rightarrow \neg m$ et $r_4^k \in$ BC de la forme $r_4^k : \neg m \wedge A_4^k \rightarrow B_4^k\}$

F3 = Res(BC, Cons-Cons, "m") =$\{ r_{1,2}^{j,k}: A_1^j \wedge A_2^k \rightarrow \perp$, avec

$r_1^j \in$ BC de la forme $r_1^j : A_1^j \rightarrow m$ et $r_2^k \in$ BC de la forme $r_2^k : A_2^k \rightarrow \neg m\}$

On va montrer que

BC a un modèle $\Leftrightarrow$ BC $\cup$ (F1 $\cup$ F2 $\cup$ F3) - (E1 $\cup$ E2 $\cup$ E3 $\cup$ E4) a un modèle

$\Rightarrow$:

Soit i l'interprétation de BC.

On discute suivant les valeurs de $vv_i(m)$:

* soit $vv_i(m) = V$, d'où

a) $\forall j$, $vv_i(Cons(r_2^j)) = F \quad \Rightarrow \forall j$, $vv_i(Prem(r_2^j)) = vv_i(A_2^j) = F$ / I

b) $\forall j$, $vv_i(r_3^j) = vv_i(m \wedge A_3^j \rightarrow B_3^j) = vv_i(A_3^j \rightarrow B_3^j) = V$

Donc 1°) $\forall j,k$ $vv_i(Prem(r_{2,4}^{j,k})) = vv_i(A_2^j \wedge A_4^k) = F$ / I $\Rightarrow vv_i(r_{2,4}^{j,k}) = V$

donc $\forall r \in F2 \Rightarrow vv_i(r) = V$

2°) $\forall j,k$ $vv_i(Prem(r_{1,2}^{j,k})) = vv_i(A_1^j \wedge A_2^k) = F$ / I $\Rightarrow vv_i(r_{1,2}^{j,k}) = V$

donc $\forall r \in F3 \Rightarrow vv_i(r) = V$

Raisonnons maintenant suivant les valeurs de $vv_i(A_1^j)$.
On distingue les trois ensembles d'indices suivants:

$J = \{j, r_1^j \in E1\}$ et

$J_+ = \{j, r_1^j \in E1 , r_1^j : A_1^j \rightarrow m$ et $vv_i(A_1^j) = V\}$ et

$J_- = \{j, r_1^j \in E1 , r_1^j : A_1^j \rightarrow m$ et $vv_i(A_1^j) = F / I\}$ (on a donc $J_+ \cup J_- = J$)

On va regarder les valeurs de vérité des règles construites à partir des règles indicées par les éléments de J_+ et J_- respectivement:

$- \forall j \in J_+ , \forall k, vv_i(A_1^j \wedge A_3^k) = vv_i(A_3^k)$

donc $\forall j \in J_+ , \forall k, vv_i(r_{1,3}^{j,k}) = vv_i(A_1^j \wedge A_3^k \rightarrow B_3^k) = vv_i(A_3^k \rightarrow B_3^k) = V$

$- \forall j \in J_- , \forall k, vv_i(A_1^j \wedge A_3^k) = F / I$

donc $\forall j \in J_- , \forall k, vv_i(Prem(r_{1,3}^{j,k})) = F / I \Rightarrow vv_i(r_{1,3}^{j,k}) = V$

donc $\forall j \in J, \forall k, vv_i(Prem(r_{1,3}^{j,k})) = V$

donc $\forall r \in F1 \Rightarrow vv_i(r) = V$

donc BC $\cup$ (F1 $\cup$ F2 $\cup$ F3) - (E1 $\cup$ E2 $\cup$ E3 $\cup$ E4) a un modèle

* soit $vv_i(m) = F$, d'où

a) $\forall j, vv_i(Cons(r_1^j)) = F \quad \Rightarrow \forall j, vv_i(Prem(r_1^j)) = vv_i(A_1^j) = F / I$

b) $\forall j, vv_i(r_4^j) = vv_i(A_4^j \rightarrow B_4^j) = V$

Donc 1°) $\forall j,k \; vv_i(Prem(r_{1,3}^{j,k})) = vv_i(A_1^j \wedge A_3^k) = F / I \Rightarrow vv_i(r_{1,3}^{j,k}) = V$

donc $\forall r \in F1 \Rightarrow vv_i(r) = V$

2°) $\forall j,k \; vv_i(Prem(r_{1,2}^{j,k})) = vv_i(A_1^j \wedge A_2^k) = F / I \Rightarrow vv_i(r_{1,2}^{j,k}) = V$

donc $\forall r \in F3 \Rightarrow vv_i(r) = V$

Raisonnons maintenant suivant les valeurs de $vv_i(A_2^j)$

Posons $J = \{j, r_2^j \in E2\}$ et

$J_+ = \{j, r_2^j \in E2, r_2^j : A_2^j \rightarrow m$ et $vv_i(A_2^j) = V\}$ et

$J_- = \{j, r_2^j \in E2, r_2^j : A_2^j \rightarrow m$ et $vv_i(A_2^j) = F / I\}$ (on a donc $J_+ \cup J_- = J$)

On va regarder les valeurs de vérité des règles construites à partir des règles indicées par les éléments de J_+ et J_- respectivement:

$- \forall j \in J_+ , \forall k, vv_i(A_2^j \wedge A_4^k) = vv_i(A_4^k)$

donc $\forall j \in J_+$, $\forall k$, $vv_i(r_{2,4}{}^{j,k}) = vv_i(A_2{}^j \wedge A_4{}^k \rightarrow B_4{}^k) = vv_i(A_4{}^k \rightarrow B_4{}^k) = V$

- $\forall j \in J_-$, $\forall k$, $vv_i(A_2{}^j \wedge A_4{}^k) = F / I$

donc $\forall j \in J_-$, $\forall k$, $vv_i(Prem(r_{2,4}{}^{j,k})) = F / I \Rightarrow vv_i(r_{2,4}{}^{j,k}) = V$

donc $\forall j \in J$, $\forall k$, $vv_i(Prem(r_{2,4}{}^{j,k})) = V$
donc $\forall r \in F2 \Rightarrow vv_i(r) = V$

donc BC $\cup$ (F1 $\cup$ F2 $\cup$ F3) - (E1 $\cup$ E2 $\cup$ E3 $\cup$ E4) a un modèle

* soit $vv_i(m) = I$, d'où
  a) $\forall j$, $vv_i(Cons(r_1{}^j)) = I \quad \Rightarrow \forall j$, $vv_i(Prem(r_2{}^j)) = vv_i(A_1{}^j) = F / I$
  b) $\forall j$, $vv_i(Cons(r_2{}^j)) = I \quad \Rightarrow \forall j$, $vv_i(Prem(r_2{}^j)) = vv_i(A_2{}^j) = F / I$

Donc $\forall j,k \quad vv_i(Prem(r_{1,3}{}^{j,k})) = vv_i(Prem(r_{2,4}{}^{j,k})) = vv_i(Prem(r_{1,2}{}^{j,k})) = F / I$
$\Rightarrow \forall j,k \quad vv_i(r_{1,3}{}^{j,k}) = vv_i(r_{2,4}{}^{j,k}) = vv_i(r_{1,2}{}^{j,k}) = V$
$\Rightarrow \forall r \in F1 \cup F2 \cup F3, \quad vv_i(r) = V$

donc BC $\cup$ (F1 $\cup$ F2 $\cup$ F3) - (E1 $\cup$ E2 $\cup$ E3 $\cup$ E4) a un modèle

donc dans tous les cas si BC a un modèle,
BC $\cup$ (F1 $\cup$ F2 $\cup$ F3) - (E1 $\cup$ E2 $\cup$ E3 $\cup$ E4) a un modèle

$\Leftarrow$:
On sait que $m \notin (MO_{BC'} \cup E)$. Donc si i représente l'interprétation de BC', i(m) n'est pas défini.
On va discuter de la valeur à donner à $vv_i(m)$ en fonction des valeurs des $vv_i(A_j{}^k)$ pour que i devienne un modèle de BC.

* Si $\exists$ j0 tel que $vv_i(A_1{}^{j0}) = V$ avec $r_1{}^{j0} \in$ BC et $r_1{}^{j0} : A_1{}^{j0} \rightarrow m$
alors:
  a) $\forall j$, $vv_i(r_{1,3}{}^{j0,j}) = vv_i(A_1{}^{j0} \wedge A_3{}^j \rightarrow B_3{}^j) = vv_i(A_3{}^j \rightarrow B_3{}^j) = V$
  b) $\forall j$, $vv_i(Prem(r_{1,2}{}^{j0,j})) = vv_i(A_1{}^{j0} \wedge A_2{}^j) = vv_i(A_2{}^j) = F / I$.

$\Rightarrow$ **on choisit $vv_i(m) = V$.**
On a:
  1°) $vv_i(m) = V$ donc $\forall j$, $vv_i(A_1{}^j \rightarrow m) = V$ donc $\forall j$, $vv_i(r_1{}^j) = V$
  2°) $\forall j$, $vv_i(A_2{}^j) = F / I$ donc $\forall j$, $vv_i(Prem(r_2{}^j)) = F / I$ donc $\forall j$, $vv_i(r_2{}^j) = V$

3°) $\forall j,\ vv_i(A_3^j \rightarrow B_3^j) = V$ donc $\forall j,\ vv_i(r_3^j) = V$

4°) $vv_i(\neg m) = F$ donc $\forall j,\ vv_i(\neg m \wedge A_4^j) = vv_i(Prem(r_4^j)) = F$ donc $\forall j,\ vv_i(r_4^j) = V$

donc $\forall r \in E1 \cup E2 \cup E3 \cup E4,\ vv_i(r) = V$.

* Sinon, (ie $\forall j,\ vv_i(A_1^j) = F$ / I)

* Si $\exists$ j0 tel que $vv_i(A_2^{j0}) = V$ avec $r_2^{j0} \in BC$ et $r_2^{j0} : A_2^{j0} \rightarrow \neg m$

Alors:

a) $\forall j,\ vv_i(r_{2,4}^{j0,j}) = vv_i(A_2^{j0} \wedge A_4^j \rightarrow B_4^j) = vv_i(A_4^j \rightarrow B_4^j) = V$

b) $\forall j,\ vv_i(Prem(r_{1,2}^{j,j0})) = vv_i(A_1^j \wedge A_2^{j0}) = vv_i(A_1^j) = F$ / I

$\Rightarrow$ on choisit $vv_i(m) = F$.

On a:

1°) $\forall j,\ vv_i(A_1^j) = F$ / I donc $\forall j,\ vv_i(Prem(r_1^j)) = F$ / I donc $\forall j,\ vv_i(r_1^j) = V$

2°) $vv_i(\neg m) = V$ donc $\forall j,\ vv_i(A_2^j \rightarrow \neg m) = V$ donc $\forall j,\ vv_i(r_2^j) = V$

3°) $vv_i(m) = F$ donc $\forall j,\ vv_i(m \wedge A_3^j) = vv_i(Prem(r_3^j)) = F$

donc $\forall j,\ vv_i(r_3^j) = V$

4°) $\forall j,\ vv_i(A_4^j \rightarrow B_4^j) = V$ donc $\forall j,\ vv_i(r_4^j) = V$

donc $\forall r \in E1 \cup E2 \cup E3 \cup E4,\ vv_i(r) = V$.

* Sinon (ie $\forall j,\ vv_i(A_2^j) = F$ / I)

$\Rightarrow$ on choisit $vv_i(m) = I$.

On a:

1°) $\forall j,\ vv_i(A_1^j) = F$ / I donc $\forall j,\ vv_i(Prem(r_1^j)) = F$ / I donc $\forall j,\ vv_i(r_1^j) = V$

2°) $\forall j,\ vv_i(A_2^j) = F$ / I donc $\forall j,\ vv_i(Prem(r_2^j)) = F$ / I donc $\forall j,\ vv_i(r_2^j) = V$

3°) $vv_i(m) = I$ donc $\forall j,\ vv_i(m \wedge A_3^j) = vv_i(Prem(r_3^j)) = F$ / I

donc $\forall j,\ vv_i(r_3^j) = V$

4°) $vv_i(\neg m) = I$ donc $\forall j,\ vv_i(\neg m \wedge A_4^j) = vv_i(Prem(r_4^j)) = F$ / I

donc $\forall j,\ vv_i(r_4^j) = V$

donc $\forall r \in E1 \cup E2 \cup E3 \cup E4,\ vv_i(r) = V$.

Donc, dans tous les cas, le modèle de BC' s'étend facilement à BC. Donc les deux ensembles de règles BC et BC' sont équivalents au regard de la cohérence (par rapport à un ensemble E tq $E \cap MD_{BC} = \emptyset$).

CHAPITRE IV: L'outil MELOMIDIA

1) Introduction

Le travail qui va être décrit ici consiste principalement en la mise en pratique des algorithmes décrits au chapitre précédent sur un cas concret, à savoir les bases de règles écrites dans le cadre du projet 3-SE (cf [ANC87a], [ANC87b]) et [CHERIA90]). Toutefois, des besoins supplémentaires étant apparus lors du développement de ce projet, des outils complémentaires à celui de vérification de la cohérence ont été également développé. Nous présenterons ici ceux pouvant s'avérer également utiles pour d'autres bases de connaissances.

L'outil MELOMIDIA (pour MEthodes en LOgique MInimale de Détection d'Incohérence et d'Anomalies), tel qu'il se présente aujourd'hui, est constitué d'une chaîne de modules destinés à:

- la détection de contradictions et la construction d'un filtre de bases de faits initiale (cf 3) et 4)c)),
- la recherche des processus de contradiction (4) b))
- l'optimisation des bases de connaissances (cf 2)d), 3) et 4)a))

Par ailleurs, le résultat issu du traducteur pourrait très bien être utilisé pour améliorer le mécanisme d'exploitation des bases de connaissances (aspect "compilation", cf 2)d)).

Il n'est pas de notre propos ici de donner les spécifications précises des programmes développés. Le lecteur intéressé pourra s'il le désire consulter le dossier technique des programmes présenté dans [LAFO&T90]. Nous passerons donc rapidement, et seulement à titre d'exemple, sur les parties relevant de l'implémentation des algorithmes exposés au chapitre 2. Nous nous attarderons davantage sur les fonctionnalités nouvelles. Puis nous exposerons les résultats obtenus. Enfin, une comparaison avec le programme MELODIA (cf [CHARLES90]), point de départ de notre recherche dans le domaine, sera effectuée.

2) La Traduction/ Compilation en "clauses"

Le but de cette partie est de traduire une base de règles dans la syntaxe décrite au chapitre II 3) b), ie en un ensemble de productions. On sait que celles-ci proviennent soit directement des règles, soit de la réécriture sous forme de productions des contraintes logiques générées par la traduction. On distinguera donc ici aussi l'obtention de ces deux ensembles.

De plus, par rapport à la théorie exposée au chapitre II, et dans le souci de diminuer les temps de lecture de la base de règles traduite, **on codera chaque expression (entité comparateur valeur) de la base de règles un entier relatif.** Les ensembles de mots que nous considérons seront donc ici des ensembles de nombres.

a) La syntaxe GENESIA1-TR

Les règles à analyser sont de la forme:
REGLE <nom de règle> SI <Partie-Prémisse> ALORS <Partie-Conséquent>

- La partie prémisse étant une conjonction de prémisses, une prémisse étant un expression, ou une disjonction d'expressions, de la forme:
<entité> <Comparateur><valeur>
Quand la prémisse est une disjonction d'expressions, on parle de prémisse disjonctive.

- La partie conséquent étant une conjonction d'expressions de la forme
<entité> = <valeur>
une <valeur> étant un nombre ou une chaîne de caractères.
On n'a donc pas d'expression liant entr'elles plusieurs entités (du type E = F + G).

b) Obtention de l'ensemble des productions directement issues des règles

Afin d'obtenir des productions du type indiqué au chap II, il est nécessaire de transformer les règles en expressions plus élémentaires, c'est-à-dire en règles ne possédant pas de disjonction en partie prémisse et n'ayant qu'un seul conséquent. Par ailleurs, dans le but de simplifier notre algèbre d'intervalle, nous allons réécrire toutes les propositions à l'aide des trois relations élémentaires ">", "<" et "=".

Plus précisément,

1°) On commence, pour minimiser le nombre de comparateurs à gérer, par réécrire:
- toutes les expressions de la forme $X \geq Y$ sous la forme $X > Y$ ou $X = Y$.
- toutes les expressions de la forme $X \leq Y$ sous la forme $X < Y$ ou $X = Y$.
- toutes les expressions de la forme $X \neq Y$ sous la forme $X > Y$ ou $X < Y$[1].

2°) On associe à chaque expression un entier relatif, dans l'ordre d'apparition dans la base de règles.

3°) On remplace une règle (notée R) ayant une prémisse (notée P) de la forme :
A_1 ou A_2 ou ... ou A_p
par p règles R_i telles que:
$Prem(R_i) = (Prem(R) - \{P\}) \cup \{A_i\}$; éclatement de la disjonction;
$Cons(R_i) = Cons(R)$; même partie conséquent;

Cette opération est possible puisqu'avec notre syntaxe, les conséquents sont indépendants des prémisses:
on n'a pas de conséquent du type F = E, avec E une entité testée dans une prémisse.

---

[1] Signalons le cas particulier de l'expression $X \neq Y$ avec Y une chaîne de caractères qui sera réécrite $\neg$ (E = e) si e est une chaine de caractères, la sémantique de "$\neg$" étant celle exposée au chapitre précédent.
Les deux traductions sont équivalentes sur le plan de la recherche de la cohérence. La deuxième est plus pratique pour filtrer les bases de faits et restituer à l'expert les résultats obtenus.

4°) On remplace une règle ayant n conséquents par n règles ayant toutes la même partie prémisse (celle de la règle initiale), et chacune un des conséquents de la règle initiale: si Cons(R) = C_1 et C_2 et ... et C_n , on créée n règles définies par
Prem(R_i) = Prem(R) et Cons(R_i) = C_i

Cette opération est légitime, puisque en raison de la syntaxe, tous les conséquents d'une règle sont indépendants: on ne peut avoir par exemple les deux conséquents (inséparables): E = F + G et H = E + 1.

5°) Enfin, on marque d'une étoile l'entier représentant le conséquent de la règle considéré. A chaque règle correspond donc un ensemble de suites d'entiers relatifs, le dernier élément de cette suite étant étoilé.

Exemple:
Supposons qu'une base de règles soit constituée des deux règles:
REGLE UN
SI E1 ≠ 1
E2 ≥ 3
ALORS E3 = 5
E5 = 'vrai'

REGLE DEUX
SI E5 ≠ 'faux'
ALORS E3 = 4

- On duplique E ≠ 1 en E > 1 ou E < 1 et E2 ≥ 3 en E > 3 ou E = 3

- On fait les associations nombre/expression suivantes:

les nombres	1 et 2	correspondent respectivement à:	E1 < 1 et E1 > 1
les nombres	3 et 4	correspondent respectivement à:	E2 > 3 et E2 =3
les nombres	5 et 8	correspondent respectivement à:	E3 = 5 et E3 = 4
les nombres	6 et 7	correspondent respectivement à:	E5 = 'vrai' et E5 = 'faux'

- et on associe à la base de règles la réunion des deux ensembles suivants:
- suites issues de la règle UN
1 3 5*
2 3 5*
1 4 5*
2 4 5*
1 3 6*
2 3 6*
1 4 6*
2 4 6*
- suite issue de la règle DEUX
-7[2] 8*

---

[2] On note -l l'expression logique ¬l. Ici -7 signifie ¬7

c) Obtention de l'ensemble des contraintes logiques, et des productions associées à ces contraintes

Si l'on veut que la traduction soit "complète", ie que l'on puisse faire les mêmes déductions dans le système issu de la traduction que dans le système initial, il manque à l'ensemble des productions issues des règles les relations entre mots correspondant aux relations entre expressions qu'ils représentent. On appelle ces relations les contraintes logiques associés à la traduction. Elles sont obtenues grâce aux tableau ci-dessous.

Plus précisément, si l1 représente l'expression (E comp1 v1) et l2 l'expression (E comp2 v2), avec v1 ≠ v2 ou comp1 ≠ comp2, on a les relations suivantes:
(le tableau étant symétrique, on ne remplit que les cases de la moitié inférieure)

l1 \ l2	**E > v2**	**E < v2**	**E = v2**
E > v1	si v1 > v2 , l1 ⇒ l2 si v1 < v2 , l2 ⇒ l1		
E < v1	si v1 > v2 , ¬ l1 ⇒ l2 si v1 ≤ v2 , l1 ⇒ ¬l2	si v1 > v2 , l2 ⇒ l1 si v1 < v2 , l1 ⇒ l2	
E = v1	si v1 > v2 , l1 ⇒ l2 si v1 ≤ v2 , l1 ⇒ ¬l2	si v1 ≥ v2 , l1 ⇒ ¬l2 si v1 < v2 , l1 ⇒ l2	l1 ⇒ ¬l2

Les relations ci-dessus sont des implications logiques. Pour pouvoir ne considérer que des productions, on réécrit les contraintes logiques en appliquant l'algorithme de réécriture des contraintes en productions décrit au chapitre II. Puis on réécrit sous forme de suite de nombres, comme au paragraphe a.2).

Par exemple, dans l'ensemble précédent, on rajoute les implications:

5 ⇒ ¬8 c'est-à-dire E3 = 5 ⇒ ¬(E3 = 4)

6 ⇒ ¬7 c'est-à-dire E5 = 'faux' ⇒ ¬(E5 = 'vrai')
qu'on réécrit sous la forme:

5 ∧ 8 → ⊥ correspondant à la règle: SI E3 = 5 et E3 = 4 ALORS ⊥ (règle 3 de l'algo.)

6 → -7 correspondant à la règle: SI E5 = 'vrai ALORS E5 ≠ 'faux' (règle 2 de l'algo.)
et qui sont mémorisée sous la forme 5 8 et 6 -7*

d) Résultat de la Traduction. Notations propres à ce chapitre

On appelera, par analogie avec la logique des propositions:
- une variable: chaque nombre apparaissant dans une des suites de nombres produites par le traducteur, c'est-à-dire ce que nous avons appelé un mot jusqu'ici.
- une clause: chaque suite de nombres du type de celles ci-dessus
- une clause-règle: chaque clause directement issue d'une règle. C'est-à-dire toutes les clauses produites dans le module décrit au b).
- une clause d'exclusion: chaque clause provenant de la réécriture sous forme de production(s) d'une contrainte logique. C'est-à-dire toutes les clauses produites dans le module décrit au b).

On parlera donc de l'ensemble des clauses-règles pour désigner l'ensemble des productions issues directement des règles, et de l'ensemble des clauses d'exclusion pour désigner l'autre ensemble de clauses.

En résumé, le traducteur fournit donc, à partir de la base de règles, les ensembles suivants :

- **L'ensemble des clauses**, qui est la réunion de l'ensemble des clauses-règles et de l'ensemble des clauses d'exclusion. Cet ensemble correspond à l'ensemble des productions, encore appelées ensemble des règles-axiomes, dans les chapitres II et III.

- **L'ensemble des variables**, avec leur signification, ie avec la correspondance: variable <----> (entité comparateur valeur).

- **L'ensemble des variables non déductibles** et **l'ensemble des variables déductibles.**
Une variable déductible est une variable associée à une expression (ent comp valeur) dans laquelle ent est une entité déductible, ie présente au moins une fois dans un conséquent de règle. C'est que l'on a appelé jusqu'ici un mot déductible (ie un élément de MD_{BC})
Une variable non déductible est une variable associée à une expression (ent comp valeur) avec ent une entité non déductible (ie jamais présente en conséquent de règle). C'est que l'on a appelé jusqu'ici un mot non déductible ou plus souvent un mot non récepteur (ie un élément de MNR_{BC})

e) La Détection des règles trivialement inutiles

Enfin, deux dernières phases complètent le traitement avant la recherche de la cohérence. Ces deux modules vont successivement:
1°) a) **Ordonner les littéraux**, par valeur absolue, dans les clauses.
Par exemple, la clause: -8 6 -7 20
sera réécrite: 6 -7 -8 20

b) **Détecter,** lors de l'opération ci-dessus, **un premier type de règle inutile.**
En effet, on élimine les clauses présentant le schéma: X y Z -y T
ou le schéma: X y Z y* , avec X, Z et T des suites de nombres et Y un nombre
en effet, ces clauses représentent respectivement une règle
- contenant une prémisse et son opposé, par exemple E = 'oui' et E ≠ 'oui'
- concluant sur une de ces prémisses. Elle sont donc inutiles pour l'inférence.

2°) **Epurer la base de règles, en procédant à l'opération d'élimination des subsomptions,** (voir partie théorique)

- d'abord au sein de l'ensemble des clauses-règle. Ce qui revient à éliminer:

+ les redondances entre clauses. On élimine la "clause-règle"
R1: Si A et B Alors c,
si on possède la "clause-règle"
R2: Si A Alors c
+ les clauses correspondant à des règles indéclenchables si l'on se place en logique monotone (ie si l'on fait l'hypothèse de la cohérence de la base de règle).
Plus précisément, on élimine la règle R1: Si A et b Alors D,
si on possède la règle R2: Si E Alors ¬b , avec A ⊃ E

- puis on élimine les clauses-règles subsumées par une clause d'exclusion[3]:

une clause-règle qui contient une clause d'exclusion correspond à une règle qui contient deux prémisses exclusives.
En effet, dans notre syntaxe, deux prémisses logiquement exclusives sont deux prémisses correspondant à deux tests sur une même entité non simultanément satisfiables. Or, les rapports d'exclusion ou de conséquence entre expressions distinctes construites sur une même entité sont tous consignés dans les clauses d'exclusion associées à cette entité. Donc l'exclusion entre les deux tests se retrouve forcément dans l'une des clauses d'exclusion associées à l'entité.
Par exemple, supposons qu'une règle contienne les deux prémisses
E > 1 et E = 0, codées l1 et l0. Sa représentation clausale est donc de la forme: X l1 Y l0 Z
On sait, d'après le tableau du 2)c), qu'on a alors l'implication: l0 ⇒ ¬l1
Donc on a (au moins) à l'issue de l'algorithme de réécriture des implications en règles, l'une des trois clauses d'exclusion suivantes:
l0 -l1* (règle un), l1 -l0* (règle deux) ou l0 l1 (règle trois).
Ces trois clauses subsument chacune X l1 Y l0 Z. Donc la clause-règles est nécessairement subsumée par l'une des clauses d'exclusion associées à l'entité E.

On a donc, à l'issue de ces traitements, un ensemble de clauses épuré des clauses trivialement inutiles, et dans lequel les liens entre expressions logiques ont étés, une fois pour toutes, calculés et écrits "en dur". Ce genre d'opérations permet de diminuer considérablement la complexité du moteur d'inférence qui exploite ces bases de connaissances, diminue l'espace nécessaire au stockage et le temps de lecture de la base de règles traduite. On peut donc parler de compilation, ce traitement étant d'ailleurs très proche des premières opérations effectuées par F. DANG-TRAN dans sa compilation de bases de connaissances LRC (cf ([DANG&LA88]).

---

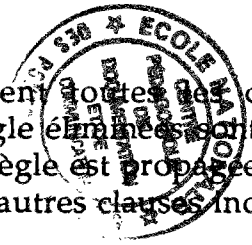
[3] L'ensemble des clauses d'exclusion ne peut contenir, par construction, de subsomptions.

3) Les Vérificateurs de Déclenchabilité et de Cohérence

Le module principal de MELOMIDIA est bien sur, celui de la recherche de cohérence. Comme on l'a vu dans le chapitre théorique, cette recherche permet également de détecter des clauses correspondant à des règles indéclenchables. Toutefois, dans le programme de cohérence, cette recherche se faisant parallèlement à celle des règles annexes (ie ne pouvant participer à des démonstrations contradictoires), il y a un risque qu'une clause-règle ne soit pas détectée comme étant indéclenchable parce qu'elle a été préalablement éliminée comme étant annexe. Pour éviter ce cas de figure, on a extrait du programme de cohérence le module détectant les règles indéclenchables, de façon à pouvoir le faire fonctionner indépendamment.

Nous allons maintenant présenter séparément les deux modules.

a) Détection dynamique des clauses correspondant à des règles inatteignables



L'algorithme élimine systématiquement toutes les clauses dont la partie prémisse est indémontrable. Les clauses-règle éliminées sont dite inatteignables, ou indéclenchables. L'élimination des clauses-règle est propagée dans le système. Cette propagation peut conduire à la détection d'autres clauses indéclenchables.

Les clauses éliminées sont stockées de façon à pouvoir :

1°) les présenter, une fois décompilées, à l'utilisateur

2°) Reconstruire la partie utile, c'est-à-dire celle contenant le sous-ensemble maximal de l'ensemble des clauses ne contenant que des clauses-règle déclenchables.

Par exemple, si on a les règles:

REGLE UN
SI A = 1 ALORS E = 10

REGLE DEUX
SI B = 1 ALORS E = 50

REGLE TROIS
SI E > 100 ALORS F = 50

REGLE QUATRE
SI F > 10 ALORS G = 1

REGLE CINQ
SI A = 2 ALORS G = 2

Le vérificateur de déclenchabilité de MELOMIDIA signalera les règles TROIS et QUATRE comme étant indéclenchables.

Le traitement est dynamique: on examine immédiatement quelles sont les conséquences de l'indéclenchabilité d'une ou plusieurs règles sur la déclenchabilité des autres. Sur la base ci-dessus, on a d'abord détecté que le test E > 100 ne pouvait être satisfait, étant données les valeurs que l'on attribue à E (10 et 50). On élimine donc la règle TROIS. Cette élimination rend le littéral F > 10 à son tour non atteignable. Donc la règle QUATRE devient indéclenchable.

Remarque: En raison de l'autre cas d'élimination, MELOMIDIA ne trouve pas toutes les règles indéclenchables d'une base de règles. En effet, certaines peuvent être éliminées en tant que règles annexes avant d'être détectées indéclenchables.

b) La Construction des Conjectures d'Incohérence

Les clauses restantes à l'issue de la recherche de la BC-Cohérence sont appelées des conjectures d'incohérence. On a vu que chacune d'elles correspond à un ensemble de conditions minimales qu'une base de faits initiale doit satisfaire pour qu'elle permette, lors du processus d'inférence avec la base de règles concernée, de déduire des faits contradictoires.

Cet ensemble, s'il est non-vide, est retraduit en clair puis présenté à l'utilisateur. Celui-ci pourra s'en servir comme entrée des modules de traçage et de filtrage de la base de faits initiale (voir la partie suivante).

Exemple: si on reprend la base de règles du 1)

REGLE UN
SI E1 ≠ 1
E2 ≥ 3
ALORS E3 = 5
E5 = 'vrai'

REGLE DEUX
SI E5 ≠ 'faux'
ALORS E3 = 4

MELOMIDIA fournit les spécifications de bases de faits initiales suivantes:

E1 > 1, E2 > 3 (2 3)
E1 < 1, E2 > 3 (1 3)
E1 > 1, E2 = 3 (2 4)
E1 < 1, E2 = 3 (1 4)

Note: l'algorithme continue et élimine la règle CINQ qui est devenue annexe puique ayant un conséquent non contredit et ne servant pas à satisfaire une prémisse d'une autre règle. Puis, par coupure obtient la spécification de la conjoncture d'incohérence: A = 1, B = 1

4) Les Outils annexes

Ces outils prolongent les résultats fournis par les parties précédentes. On a construit:

a) Le Reconstructeur de la partie utile de la base de règles

Ce module opère la réunion des trois ensembles de clauses correspondant à des règles indéclenchables détectées lors de la traduction et de la vérification de la cohérence, soustrait l'ensemble obtenu de l'ensemble des clauses, et reconstruit la base de règles correspondante. On obtient ainsi la base de règles réellement utile pour le processus d'inférence.

Dans son état actuel, ce module ne reconstitue pas les règles ayant été éclatées lors de la phase de traduction (voir 2)b)). Il ne fournit donc que des règles élémentaires, c'est-à-dire n'ayant pas de OU en prémisse et ne possédant qu'un seul conséquent. Cette restriction sur les règles pénalise de manière insignifiante les performances du moteur au moment de l'exploitation de la base de règles. Bien entendu, elle ne modifie nullement le résultat de l'inférence. Par contre, elle fausse les chiffres comptabilisant le nombre de règles avant et après extraction des règles inutiles et permettant d'apprécier le gain obtenu.

Enfin, se module fournit également en clair, ie après décompilation, les clauses-règles détectées indéclenchables.

b) Le Traceur-Démonstrateur d'incohérences

Cet utilitaire est un moteur d'inférence un peu modifié de façon à ce qu'il s'arrête à la première contradiction effectuée, et auquel on a associé un mécanisme de trace précis. Il prend en entrée une conjecture d'incohérence ainsi que la base de règles à étudier, et fournit en sortie la démonstration (contradictoire) obtenue en effectuant un cycle d'inférences en prenant cette conjecture comme base de faits initiale et comme base de règles la base étudiée.

Ce module a été réalisé pour combler une des lacunes de la méthode, à savoir de fournir les conjectures d'incohérences mais pas ce qui est contredit, ni bien sur comment on obtient cette contradiction. On a ainsi un moyen de connaître le processus de déduction de conclusions contraires, sans que la recherche de la cohérence ne soit pénalisé par la mémorisation, pour chaque conjecture, des étapes effectuées pour l'obtenir. Cette fonctionnalité est apparue indispensable pour corriger les grosses bases de règles sur lesquelles nous travaillions.

c) Le Filtreur de bases de faits initiales

Ce dernier utilitaire est issu directement de notre contexte de travail, mais peut aussi se rencontrer ailleurs. En effet, les bases de règles sur lesquelles nous travaillons étant générées automatiquement, on savait (postulat) qu'elles ne pouvaient contenir d'erreurs.

Par contre, elles ne sont pas conçues pour traiter n'importe quelle base de faits initiale. Le but de la recherche de la cohérence était donc principalement d'obtenir la liste exhaustive des spécifications des bases de faits initiales pouvant conduire à des déductions contradictoires.

Une fois cette liste construite, il faut donc l'utiliser pour filtrer les bases de faits initiales se présentant en entrée de la base de règles contrôlée. Ce module réalise ce filtrage. Il prend en entrée les conjonctures d'incohérences (pas forcément toutes, mais celles que l'utilisateur a sélectionnées), et la base de faits initiale à contrôler, et se prononce sur la validité de la base de ces faits en regard de ces contraintes.

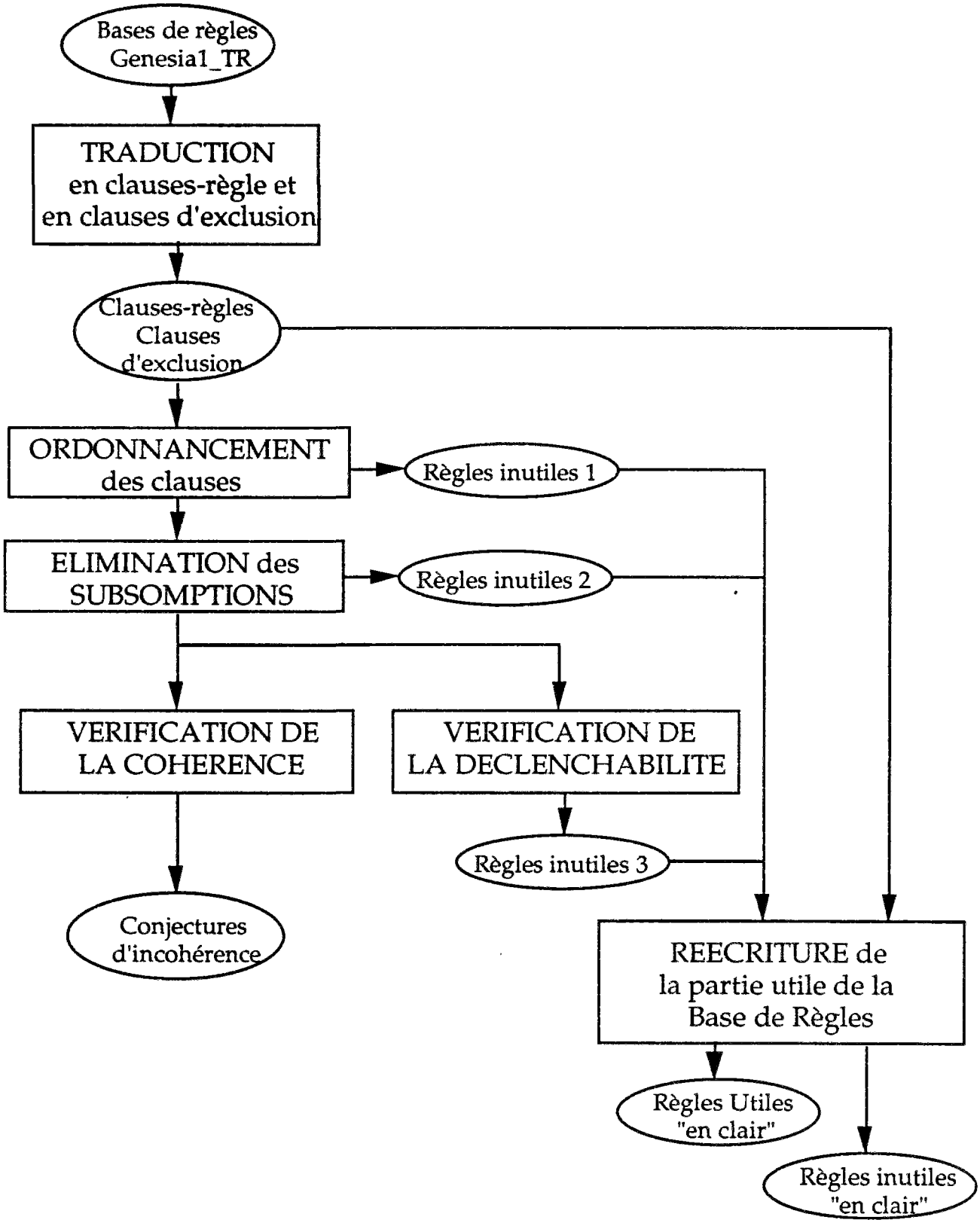
Par exemple, si la base de faits à contrôler est { (A, 1), (E , 5), (F, 4), (G, 8)},
et que l'ensemble des contraintes contient la contrainte: E > 2 et F < 5
il fournit cette contrainte[4].

On se reportera à [LAINE91] pour plus de précisions sur les premiers et troisièmes modules décrits ici.

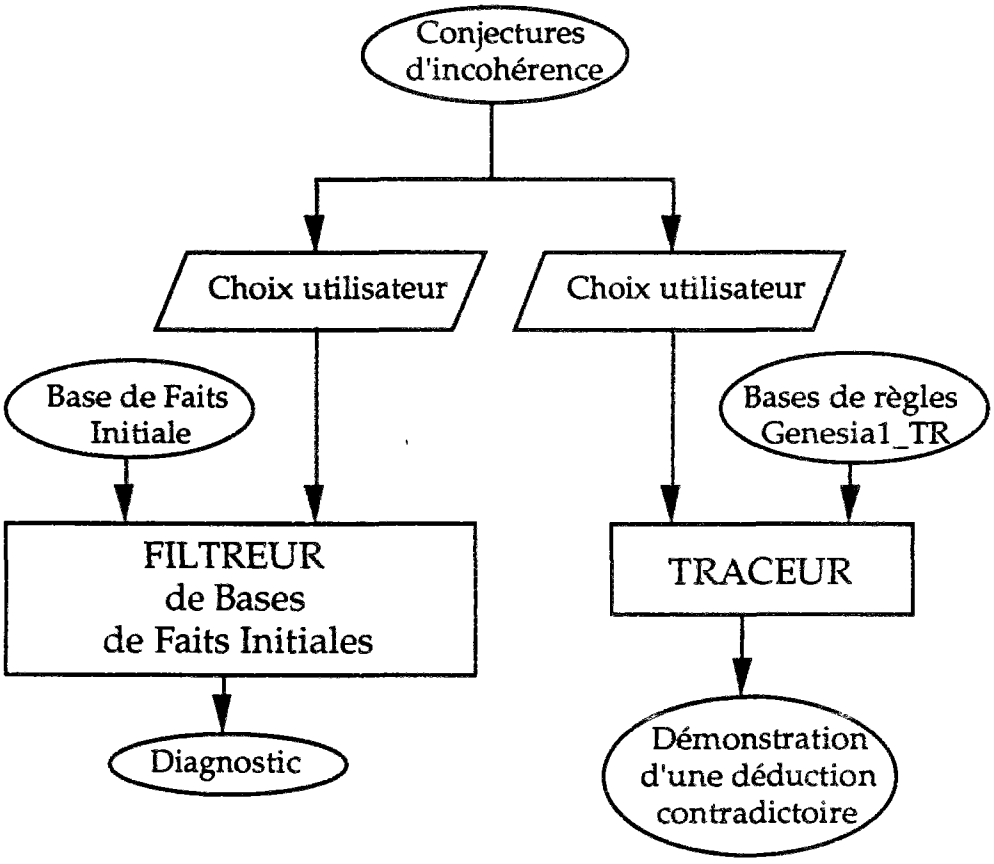
[4] Ce n'est donc pas tout-à-fait qu'un problème trivial de recherche d'inclusions. Par ailleurs, vu le nombre de contraintes et de faits à considérer (plusieurs milliers d'éléments dans chaque ensemble), ce n'est pas si simple que cela peut paraître.

5) Synoptiques des traitements effectués par MELOMIDIA

a) Traduction, Vérification de la cohérence et de la déclenchabilité



b) Les traitements postérieurs aux vérifications de cohérence et de déclenchabilité



6) Résultats

Nous allons présenter maintenant les résultats obtenus par MELOMIDIA sur les deux bases de règles qui ont servi de jeux de test. Ces bases sont toutes les deux issues du projet 3-SE, et correspondent à son état d'avancement en 1989 et en 1991.

a) Résultats obtenus sur la base 3-SE89

a.1) Caractéristiques de la base de règles

Il s'agit d'une base de règles de reconstitution de l'état d'un tranche d'une centrale nucléaire à partir des informations fournies par des capteurs. Les entités initiales de cette base sont donc les capteurs de la tranche. La base permet de déduire l'état des matériels (pompe, vannes etc ...) de la tranche.

Les particularités de cette application sont:

- le grand nombre de règles de la base de règles (> 10 000), **bien que la base fournie à l'époque ne soit pas complète**. Il manque en effet des règles de liaison entre entités, ce qui conduit à rendre non déductibles des entités qui ne sont pas non plus initiales: ce ne sont pas des capteurs, et il manque les règles permettant de connaître leur état.

- la différence importante (facteur 4) entre le nombre d'entités initiales (une entité est initiale si elle peut être être présente en BFi) reconnue par l'expert et le nombre d'entités trouvées non déductibles par le traducteur de MELOMIDIA (une entité est non déductible si elle n'apparait jamais en conséquent de règle). Cela est du:

1°) à l'incomplétude de la base évoquée dans le point précédent,

2°) au fait que la base de règles est très générale, ie utilisable pour la simulation du comportement d'autres tranches. Elle tient donc compte de certains types de capteurs non présents dans la tranche concernée par cette application.

3°) au fait que la tranche de centrale visée n'étant pas encore totalement instrumentée, et que seules sont déclarées initiales les entités correspondant à des capteurs présents dans l'installation.

SYNOTIQUE des CARACTERISTIQUES de la base 3SE-89:	
- nombre de règles:	10562
- nombre d'entités:	8181
- nombre d'entités non déductibles:	3600
- nombre d'entités reconnues initiales par l'expert:	846
(⇒ nombre d'entités non initiales et non déductibles: 2754)	

a.2) Traduction en clauses-règles et en clauses d'exclusion

La traduction s'effectue en 5 mn CPU. Elle fournit les résultats suivants:
(on appelle variable initiale une variable associée à une expression (ent comp val) avec ent une entité initiale ie une entité non réceptrices et déclarée initiale par l'expert)

SYNOPTIQUE de la TRADUCTION de la base 3SE-89:

- nombre total de clauses:	18 322
- nombre de clauses-règle:	12 126
- nombre de clauses d'exclusion:	6 196
- nombre total de variables:	14 347
- nombre de variables initiales:	1 667
- nombre de variables déductibles:	8 011

(⇒ nombre de variables non initiales et non déductibles: 4 669)

- Temps CPU de traduction: 5mn

a.3) Ordonnancement

La procédure d'ordonnancement ne détecte ici aucune anormalité. Elle nécessite quelques secondes CPU.

a.4) Recherche des subsomptions

Le module de recherche de subsomptions élimine 120 clauses. Elles correspondent toutes à des clauses répétées deux fois dans le système.

Exemple: (extrait du listing fourni par le module)

```
la clause            7890
*  1668*  -2882          3299*
est subsumee par       18176
* 10424*  -2882          3299*
```

```
la clause   7890
*  1737*  -1824   -1830   3369*
est subsumee par     18245
* 10493*  -1824   -1830   3369*
```

Numéro d'identification de la clause

Numéro de la règle dont est issue la clause

Ces 120 clauses correspondent à 112 règles répétées deux fois dans la base. (on trouve 112 règles plutôt que 120 car 8 règles parmi ces 112 ne sont pas élémentaires et correspondent chacune à deux clauses).

a.5) Vérification de la cohérence

La vérification de la cohérence nécessite 2 mn 30 s CPU. Elle s'effectue par rapport à l'ensemble des variables initiales, et non par rapport à l'ensemble des variables/mots non récepteurs, comme dans la BC-Cohérence. En effet, on veut garantir l'inférence uniquement à partir de toute BFi composée d'entités initiales.

Cette vérification permet la détection de 9344 clauses d'incohérences[5], qui doivent ici être interprétées comme des contraintes sur la base de faits initiale.

a.6) Reconstruction de la partie utile de la base de règles

Rappelons que le reconstructeur de la partie utile de la base de règles fournit les règles élémentaires (1 seul conséquent, pas de disjonction) utiles pour l'inférence.
On appelera "proposition" une prémisse ou un conséquent. Enfin, trois taux de compression de la BR initiale, grâce à cette opération, vont être définis:

$$\text{TCOMPRESS1} = \frac{\text{Nombre initial de règles}}{\text{Nombre de règles élémentaires fournies par le reconstructeur}}$$

$$\text{TCOMPRESS2} = \frac{\text{Nombre de propositions contenues dans la BR initiale}}{\text{Nombre de propositions contenues dans la BR reconstruite}}$$

$$\text{TCOMPRESS3} = \frac{\text{Nombre de règles élémentaires du système initial}}{\text{Nombre de règles élémentaires fournies par le reconstructeur}}$$

Le premier taux de compression est le taux "brut". Il n'est pas complètement significatif puisqu'on compare de type d'objet différents, des règles élémentaires d'une part, et des règles normales d'autre part. Les deux autres taux corrigent cette approximation en fournissant les rapports d'encombrement (TCOMPRESS2) et le taux de compression pour un type de règles identiques (TCOMPRESS3).

RESULTAT de la RECONSTRUCTION de la PARTIE UTILE de la BR		
- Nombre de clauses-règles inutiles:	120	
- Nombre de clauses-règles indéclenchables:	8062	
⇒ Nombre de règles élémentaires utiles:	3944	
correspondant à:	8176	propositions
- Taux de compression:		
- TCOMPRESS1:	2,7	
- TCOMPRESS2:	3,4	
- TCOMPRESS3:	3,1	

[5] Et non 27874 clauses, comme le trouvait MELODIA. La différence provient du fait que pour pouvoir traiter cette base de règles, MELODIA s'abstenait de procéder à l'élimination des subsomptions, opération particulièrement couteuse en temps calcul, et qui garantit la minimalité du système de contraintes générés. En effet, les tentatives d'execution avec recherche de subsomption n'étaient pas achevées au bout d'une heure CPU . Les 27874 clauses découvertes par MELODIA sont en fait subsumées par les 9344 clauses fournies par MELOMIDIA.

On gagne donc approximativement un facteur trois.

a.7) Un Exemple d'incohérence: utilisation du traceur d'incohérence

Les conjectures d'incohérence sont décompilées et fournies à l'expert. Celui-ci peut vérifier alors le déroulement de la démonstration incohérente grâce au Traceur d'incohérences.

Exemple: on vérifie la conjecture d'incohérence
E2 LKB001EC = '0'; E2 LKB004EC = '0'; E2 LKB005EC = '1';
Le traceur est exécuté avec cette incohérence. Il fournit le résultat suivant

```
(*============================================================*)
 Declenchement de la règle :
REGLE 'E2LKB005EC_5040'
SI
        E2LKB005EC = '1'
ALORS
        E2LKB001JA_P := 'OV'
(*============================================================*)
 Declenchement de la règle :
REGLE 'E2LKB004EC_5050'
SI
        E2LKB004EC = '0'
ALORS
        E2LKB001TU_E := 'NO'
(*============================================================*)
 Declenchement de la règle :
REGLE 'E2LKB001EC_5050'
SI
        E2LKB001EC = '0'
ALORS
        E2LKB001TU_O := 'TP'
(*============================================================*)
 Declenchement de la règle :
REGLE 'E2LKB001TU_5080'
SI
        E2LKB001TU_E = 'NO'
ALORS
        E2LBA016JA_T := 'PR'
(*============================================================*)
 Declenchement de la règle :
REGLE 'E2LKB001TU_5100'
SI
        E2LKB001TU_O = 'TO'
    ET  E2LKB001TU_E = 'NO'
ALORS
        E2LKB_T := 'PR'
(*============================================================*)
 Declenchement de la règle :
REGLE 'E2LKB001JA_40'
SI
        E2LKB001JA_P = 'OV'
ALORS
        E2LKB001JA_T := 'AB'
```

```
(*============================================================*)
 Declenchement de la règle :
REGLE 'E2LKB001TU_680'
SI
        E2LKB001TU_E = 'NO'
    ET  E2LKB001TU_O = 'TP'
ALORS
        E2LKB_T := 'PR'
(*============================================================*)
 Declenchement de la règle :
REGLE 'E2LKB001TR_10'
SI
        E2LKB001JA_T = 'AB'
ALORS
        E2LKB001TR_T := 'AB'
(*============================================================*)
 Declenchement de la règle :
REGLE 'E2LKB_10'
SI
        E2LKB001TR_T = 'AB'
ALORS
        E2LKB_T := 'AB'
La valeur du fait E2LKB_T a change
    Par la regle E2LKB001TU_510  il avait la valeur : 'PR'
    Elle devient par la regle E2LKB_10          : 'AB'
```

Remarque: Cette démonstration est particulièrement simple. Les autres sont pour la plupart beaucoup plus compliquées (cf ANNEXE).

a.8) Filtrage de la base de faits initiale

Un prototype de vérification de la base de faits initiale a été développé par S. LAINE dans le cadre de son stage dans le groupe IA de l'EDF.
Il diagnostique en 3,5 secondes une base de 8181 faits (toutes les entités initiales sont systématiquement présentes en BFi).

Cette opération se déroule en deux phases successives:
1°) Traduction des faits zéro plus en faits zéro
Exemple : Si on a le fait A = 1, et que A est présent dans le système de règles sous la forme A > 0, A = 1 et A < 1, codés respectivement par les variables v0, v1 et v2, le fait A = 0 correspond aux faits v0, v1 et ¬v2.
C'est cette traduction qui est la plus couteuse en temps - calcul.
2°) On vérifie quelles clauses d'incohérence sont satisfaites, parmi les 9344 détectées par le programme d'incohérence, et on les signale.

Le temps actuel de diagnostic n'est pas satisfaisant, cette opération associée à celle de l'inférence proprement dite devant prendre moins de 7 secondes CPU, d'après le cahier des charges de l'application.

b) Résultats obtenus sur la base 3-SE91

b.1) Caractéristiques de la base de règles

Cette deuxième base de test correspond à une version plus récente de la même application, à savoir le diagnostic des alarmes d'une tranche d'une centrale nucléaire.

Cette fois, la base de règles est considérée complète par le concepteur, ce qui explique:

- son grand nombre de règles (> 20 000) , soit deux fois plus que la base de règles de 1989.
- sa généricité, qui explique là encore une différence importante (facteur 2,5) entre le nombre d'entités initiales reconnue par l'expert et le nombre d'entités trouvées non déductibles par le traducteur de MELOMIDIA. Cette différence est toutefois nettement moins importante que pour 3SE-89.

SYNOTIQUE des CARACTERISTIQUES de la base 3SE-91:	
- nombre de règles: 23 357	
- nombre d'entités: 15 529	
- nombre d'entités non déductibles:	3 284
- nombre d'entités reconnues initiales par l'expert:	1 340
(⇒ nombre d'entités non initiales et non déductibles:	1 944)

b.2) Traduction en clauses-règles et en clauses d'exclusion

La traduction s'effectue en 26 mn CPU. Elle fournit les résultats suivants:

SYNOPTIQUE de la TRADUCTION de la base 3SE-91:	
- nombre total de clauses:	46 181
- nombre de clauses-règle:	32 478
- nombre de clauses d'exclusion:	13 703
- nombre total de variables:	27 942
- nombre de variables initiales:	2 444
- nombre de variables déductibles:	21 825
(⇒ nombre de variables non initiales et non déductibles: 3 673)	
- Temps CPU de traduction: 26 mn	

b.3) Ordonnancement

La procédure d'ordonnancement détecte ici 176 Clauses contenant à la fois une variables et sa négation. Elles correspondent à autant de règles concluant sur une de leurs prémisses.

Exemple:
La clause-règle

```
* 17837 *       20373*     -20373     -20374
```

qui correspond à la règle:

```
REGLE E2RGL001TR_2042
SI
ET  E2RGL001TR_C = 'à'
ET  E2RGL001TR_I = 'à'
ALORS
ET  E2RGL001TR_C = 'à'
```

est une tautologie.

b.4) Recherche des subsomptions

Le module de recherche de subsomptions élimine 372 clauses. Elle nécessite 20 secondes CPU. Plus précisément, on élimine:

- 360 clauses correspondant à des clauses redondantes:

Exemple:

```
La clause       40717
*  20681*  -1705     -24141     -24832     24833*
est subsumee par        40814
*  20754*  -1705     -24141     -24832     24833*
```

Ce qui correspond aux règles

```
REGLE E2ISA_455_2/4_2ISA_451_ET_15040
SI
ET  E2ISA_451_ET_V = 'IT'
ET  E2RPR506UP_T = 'PR'
ET  E2ISA_454_ET_V = 'IT'
ET  E2ISA_453_ET_V = 'IT'
ALORS
ET  E2ISA_455_2/4_V = 'IT'

REGLE E21SA_455_2/4_2ISA_454_ET_15040
SI
ET  E2ISA_454_ET_V = 'IT'
ET  E2RPR506UP_T = 'PR'
ET  E2ISA_453_ET_V = 'IT'
ET  E2ISA_451_ET_V = 'IT'
ALORS
ET  E2ISA_455_2/4_V = 'IT'
```

Ces règles sont effectivement identiques à une permutation des variables près.

- 12 clauses car étant moins générales que d'autres clauses également présentes dans le système.

Exemple:

```
La clause       20491
*   4830* -28470     3721*      -9634
est subsumee par        29507
*  12033* -28470     3721*
```

Ce qui correspond aux règles

```
REGLE E2LNC001CS_5405
SI
ET  E2LNC001CS_A = 'RE'
ET  E2LNC001CS_T = 'PR'
ALORS
ET  E2LNC001DL_T = 'PR'

REGLE E2LNC001CS_5800
SI
ET  E2LNC001CS_A = 'RE'
ALORS
ET  E2LNC001DL_T = 'PR'
```

b.5) Vérification de la cohérence

La vérification de la cohérence n'a pu jusqu'à présent être effectuée dans sa totalité, le nombre de clauses d'incohérence générés devenant trop important pour la taille mémoire maximale que l'ordinateur nous alloue lors de l'exécution. Comme pour 3SE-89, elle s'effectue par rapport à l'ensemble des variables initiales. Les résultats qui sont fournis ici correspondent à la réponse après 1 heure de temps CPU.

Nombre de clauses d'incohérence découvertes:	18049
Nombre de variables restant dans le systèmes:	253
Nombre de clauses restant dans la système:	48 235

On perçoit donc ici les limites de notre démarche. Au bout d'une heure de temps de calcul, il reste encore 253 variables à éliminer. Par ailleurs on connait, la moyenne d'apparition dans les clauses d'une variable. Elle est à cet instant de 98 apparitions/variable. Ce qui signifie que chaque opération d'élimination d'une variable génèrera l'examen de $(98/2)^2 = 2500$ clauses, ce qui correspond, hors mis l'aspect spatial de stockage de ces nouvelles clauses, un temps de calcul de l'ordre de 20 secondes (estimation d'après les traces d'exécution, voir annexes).
Donc la résolution complète nécessiterait encore un temps de calcul de l'ordre de 250 * 20 s = 5000 secondes, soit environ 1 heure et demi de temps - calcul. Ce qui est finalement envisageable.
Nous n'avons néanmoins pas tenté l'expérience pour les raisons de stockage des clauses évoquées plus haut.

b.6) Reconstruction de la partie utile de la base de règles

Rappelons que le reconstructeur de la partie utile de la base de règles fournit les règles élémentaires (1 seul conséquent, pas de disjonctions) utiles pour l'inférence, ainsi, dans un autre fichier, que les règles élémentaires indéclenchables.

On se reportera au a.6 pour la signification des taux fournis ici.

RESULTAT de la RECONSTRUCTION de la PARTIE UTILE de la BR		
- Nombre de clauses-règles inutiles:	548	
- Nombre de clauses-règles indéclenchables:	6286	
⇒ Nombre de règles élémentaires utiles:	25 645	
correspondant à:	56 473	propositions
- Taux de compression:		
- TCOMPRESS1:	0,9	
- TCOMPRESS2:	1,2	
- TCOMPRESS3:	1,3	

Le gain est cette fois-ci beaucoup moins appréciable. Toutefois, en terme de règles élémentaires, il approche quand même de 1,5. Ce résultat est principalement du au fait qu'ici on a un système beaucoup plus proche de l'opérationnalité, donc beaucoup plus "propre".

b.7) Un Exemple d'incohérence

Parmi les 18 000 conjectures d'incohérence découvertes par MELOMIDIA, 5 ne sont composées que d'un seul littéral. On va donner ici la démonstration, particulièrement simple, associée à l'une de ces 5 conjectures. On en fournira une autre plus complexe en annexe.

Exemple: on donne la démonstration incohérente basée sur l'hypothèse initiale (ie sur la conjecture d'incohérence): E2RCV553EC = '0'

```
 (*============================================================*)
 Declenchement de la règle :
REGLE 'E2RCV553EC_5050'
SI
        E2RCV553EC = '0'
ALORS
        E2RCV003UP_T := 'AB'
(*============================================================*)
 Declenchement de la règle :
REGLE 'E2LKA718JA_5900'
SI
        E2LKA718JA_E = 'DE'
    OU  E2RCV003UP_T := 'AB'
ALORS
        E2LKA718JA_S := 'ID'
(*============================================================*)
```

```
Declenchement de la règle :
REGLE 'E2RCV257VP_6050'
SI
        E2RCV003UP_T := 'AB'
ALORS
        E2RCV257VP_S:= 'ID'
        E2RCV257VP_P:= 'FE'
(*============================================================*)
 Declenchement de la règle :
REGLE 'E2RCV251VP_6050'
SI
        E2RCV003UP_T := 'AB'
ALORS
        E2RCV251VP_S:= 'ID'
        E2RCV251VP_P:= 'FE'
(*============================================================*)
 Declenchement de la règle :
REGLE 'E2RCV259VP_6050'
SI
        E2RCV003UP_T := 'AB'
ALORS
        E2RCV259VP_P:= 'VU'
        E2RCV259VP_S:= 'ID'
(*============================================================*)
 Declenchement de la règle :
REGLE 'E2RCV026VP_6070'
SI
        E2RCV003UP_T := 'AB'
ALORS
        E2RCV026VP_P:= 'VU'
        E2RCV026VP_S:= 'ID'
(*============================================================*)
 Declenchement de la règle :
REGLE 'E2RCV026VP_6080'
SI
        E2RCV003UP_T := 'AB'
ALORS
        E2RCV026VP_P:= 'VD'
        E2RCV026VP_S:= 'ID'
La valeur du fait E2RCV026VP_P      a change
    Par la regle  E2RCV026VP_6070   il avait la valeur : 'VU'
    Elle devient par la regle  E2RCV026VP_6080  : 'VD'
```

b.8) Filtrage de la base de faits initiale

L'essai n'a pas été réalisé avec cette base de connaissances, puisque nous n'avions pas été capable de générer toutes les conjectures d'incohérence possibles. Néanmoins, étant donné que le nombre actuel de clauses générées (de l'ordre de 20000) et le nombre de variables initiales sont du même ordre de grandeur que pour 3SE-89, il est très probable de retrouver ici les mêmes résultats, ie des temps d'analyse variant entre 3 et 5 secondes.

7) Comparaison avec MELODIA

a) Introduction-Synthèse

Les recherches qui viennent d'être décrites ici sont très clairement inspirées de celles menées par E. CHARLES et qui ont abouti au logiciel MELODIA. En effet, intrigué par le fonctionnement de son algorithme, et peu convaincu de sa justesse dans le cadre qui nous intéressait, je me suis penché sur son travail et sur ses fondements théoriques. Constatant un décalage entre les postulats théoriques de MELODIA (la logique propositionnelle "pure", c'est-à-dire avec l'axiome du tiers-exclu) et la réalité (une entité peut prendre différentes valeurs ou ne pas en prendre), je repris l'algorithme de MELODIA en l'adaptant à une logique sans tiers-exclu.

Ce qui me paraissait tout d'abord être une modification mineure s'avèra en fait changer assez profondément le programme. En effet, mise à part la recherche des tautologies (appelée dans MELOMIDIA élimination des règles intrinsèquement illogiques), qui est identique dans nos deux programmes, toutes les autres opérations de MELODIA:

- la traduction en clauses (voir chapitre 2 pour le détail de la différence)
- l'élimination des subsomptions, devenue dans MELOMIDIA l'élimination des règles redondantes
- l'élimination des littéraux purs, devenue dans MELOMIDIA l'élimination des règles annexes et des règles inatteignables
- la coupure sur un mot déductible

ont été modifiées pour tenir compte de la différence des logiques sous-jacentes.

Il serait long et fastidieux de détailler formellement ces différences (le lecteur intéressé pourra se reporter au chapitre "Version révisée de l'algorithme de Davis et Putnam" du rapport [LAFON&T89]) . Nous nous contenterons de les constater en faisant fonctionner ces deux algorithmes sur une série d'exemples, les premiers purement théoriques, les suivants plus concrets.

On peut toutefois déjà donner les conclusions de cette étude.

MELODIA vérifie la cohérence d'un système formel possédant toutes les règles de déduction (l'ensemble $\mathcal{RD}$) de la logique des propositions. Cette vérification est donc plus pointilleuse que celle de MELOMIDIA qui vérifie la cohérence d'un système possédant moins de règles de déduction. On trouve donc assez facilement des "bases de règles" théoriques que MELODIA trouve incohérentes et MELOMIDIA cohérentes.

Sur le plan strict de la cohérence, c'est MELOMIDIA qui a raison. Mais il faut reconnaître que l'on ne pas donner tort (sauf exception) à MELODIA de signaler les cas en question, tant les soupçons d'anomalies sont forts (mais j'insiste, il ne s'agit que de soupçons).

En résumé, sur le plan théorique, MELODIA propose une Condition Suffisante de vérification de la BC-Cohérence et MELOMIDIA une Condition Nécessaire et

Suffisante. Une CNS est bien sur préférable, mais la condition (seulement) suffisante de MELODIA est très pertinente.

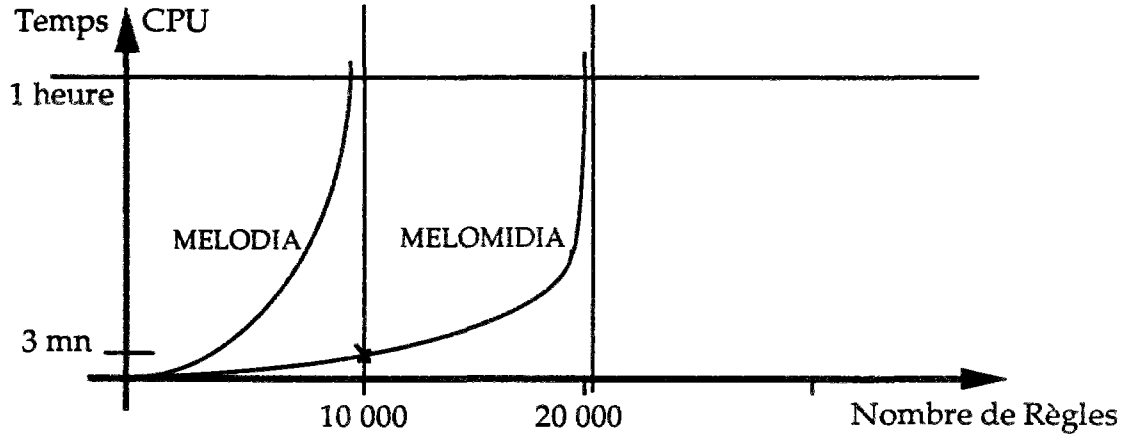
Par contre, les conjectures d'incohérence fournies par MELODIA ne sont pas minimales. On entend par là qu'elles fournissent des conditions de validité d'une base de faits initiale (cf le Filtreur de BFi, présenté au 4)) qui sont trop fortes. Autrement dit, elles peuvent empêcher de lancer une inférence avec une base de faits initiale pourtant valide. MELOMIDIA, lui, fournit les conjectures exactes.

Enfin, signalons en faveur de MELOMIDIA les points suivants:

1°) On sait précisément ce qu'il fait (on possède les démonstrations précises de toutes les étapes de l'algorithme). Donc on peut envisager sereinement d'étendre ces fonctionnalités (voir conclusion)

2°) Il constitue une chaîne opérationnelle (en libre service sur l'un des ordinateurs de l'EDF à Clamart) et complète (cf parties 2, 3 et 4 de ce chapitre). De plus, il permet la détection des règles indéclenchables qui, on l'a vu, a fourni sur nos jeux de test des résultats significatifs.

3°) Il a des performances bien meilleures que celles de MELODIA. Il est difficile d'avancer un facteur de performances dans la mesure où cette différence n'apparait que sur de très grandes bases de règles, que MELODIA ne pouvait traiter complètement, mais on a schéma de performances comparées du type:



Toutefois,

- d'une part les deux algorithmes sont clairement exponentiels,
- d'autre part la meilleure performance de MELOMIDIA est exclusivement due à une programmation plus fine (ce qui est normal puisque l'on partait du programme MELODIA dont les défauts étaient connus) et plus spécialisée (MELODIA est plus général que MELOMIDIA, bien que les différences soient à ce niveau assez mineures).

b) Exemples théoriques

On donne ici des exemples théoriques, c'est-à-dire correspondant à des ensembles de clauses ne pouvant être produit par notre traducteur à partir de bases de règles écrites dans la syntaxe GENESIA1_TR. En effet, avec cette syntaxe, il est impossible d'obtenir e dont nous avons besoin ici, à savoir que la traduction d'un conséquent de règle donne un littéral négatif (ie un littéral de la forme ¬A, avec A une variable propositionnelle). On part donc directement de bases de règles élémentaires (pas de disjonction en prémisse, un seul conséquent par règle) d'ordre zéro pur.

Dans les exemples qui vont suivre, on notera toujours:

¬X pour non X .

EIj une variable/entité initiale

A, B, ... les variables/entités non initiales

(suite page suivante)

b.1) Premier exemple : différence concernant l'opération de coupure

Soit la base de règles:

R1 : Si EI1 Alors A
R2 : Si EI2 Alors B
R3 : Si A Alors ¬B
R4 : Si ¬A Alors ¬B

à laquelle correspond le tableau suivant:

REGLES	CLAUSES MELODIA	CLAUSES-REGLES MELOMIDIA
R1 : Si EI1 Alors A	c1: ¬EI1 ∨ A	cr1: EI1 A*
R2 : Si EI2 Alors B	c2: ¬EI2 ∨ B	cr2: EI2 B*
R3 : Si A Alors ¬B	c3: ¬A ∨ ¬B	cr3: A ¬B*
R4 : Si ¬A Alors ¬B	c4: A ∨ ¬B	cr4: ¬A ¬B*

- Vérification par MELOMIDIA
> Elimination des clause-règle annexes
cr4 est éliminée puisque A est non initial est ¬A n'est pas déductible
=> cr4 est indéclenchable
> Coupure sur la variable A. On obtient le système:
cr13: EI1 ¬B*
cr2 : EI2 B*
> Coupure sur la variable B. On obtient le système:
cr122 : EI1 EI2
> Arrêt
Une BFI incohérente est une BFI contenant au moins EI1 et EI2

- Vérification par MELODIA
> Coupure sur la variable A. On obtient le système:
c13: ¬EI1 ∨ ¬B
c34: ¬B
c2 : ¬EI2 ∨ B
>Elimination des subsomptions
On supprime la clause c13 qui est subsumée par la clause c34
> Coupure sur la variable B. On obtient le système:
c2' : ¬EI2 ⇒ conjecture d'incohérence: EI2
> Arrêt
L'algorithme détecte donc bien l'incohérence, **mais ne donne pas la bonne conjecture d'incohérence.** En effet, d'après MELODIA, si une base de faits initiale contient l'entité EI2, elle entrainera des déductions contradictoires lors de l'inférence. Ce qui est trop restrictif: il faut aussi que la BFi contienne EI1pour qu'il y ait production d'incohérence.

b.2) Deuxième exemple : différence dans l'élimination des règles annexes

Soit la base de règles:

```
R1 : Si EI1 Alors A
R2 : Si EI2 et B Alors ¬A
R3 : Si ¬B  Alors B
```

à laquelle correspond le tableau suivant:

REGLES	CLAUSES MELODIA	CLAUSES-REGLES MELOMIDIA
R1: Si EI1 Alors A	c1: ¬EI1 ∨ A	cr1: EI1 A*
R2: Si EI2 et B Alors ¬A	c2: ¬EI2 ∨ ¬B ∨ ¬A	cr2: EI2 B ¬A*
R3: Si ¬B Alors B	c3: B	cr3: ¬B B*

- Vérification par MELOMIDIA
> Elimination des clause-règle annexes
cr3 est éliminé puisque B est non initial est ¬B n'est pas déductible
=> cr3 est indéclenchable
--> cr2 est éliminé puisque B est non initial est B n'est pas déductible
=> cr2 est indéclenchable
--> cr1 est éliminé puisque A est n'est pas contredit et n'est pas utilisé dans la partie prémisse d'une autre règle.
=> cr1 est inutile
=> le système est vide . Arrêt.

Pour MELOMIDIA, cette base de règles est cohérente. En effet, les BFI valides (ie contenant seulement des EI_j) n'entraînent de contradiction: seule R1 est déclenchable.

- Vérification par MELODIA
> Coupure sur la variable B. On obtient le système:
c1: ¬EI1 ∨ A
c23 : ¬EI2 ∨ ¬A
> Coupure sur la variable A. On obtient le système:
c12 : ¬EI1 ∨ ¬EI2 ⇒ conjecture d'incohérence: EI1 ∧ EI2
> Arrêt
Cet algorithme trouve la base incohérente. En regard des BFI valides, elle ne l'est pourtant pas. Si on fournit comme base de faits initiale l'ensemble {EI1, EI2}, aucune contradiction n'est inférée.

b.3) Troisième exemple : différence dans l'élimination des subsomptions

Soit la base de règles:

```
R1 : Si EI1 et EI2 Alors ¬A
R2 : Si EI1 et A Alors B
R3 : Si EI1 et A Alors ¬B
R4 : Si ¬A  Alors C
R5 : SI EI3 Alors ¬C
```

à laquelle correspond le tableau suivant:

REGLES	CLAUSES MELODIA	CLAUSES-REGLES MELOMIDIA
R1:Si EI1 et EI2 Alors ¬A	c1:¬EI1 ∨ ¬EI2 ∨ ¬A	cr1: EI1 EI2 ¬A*
R2:Si EI1 et A Alors B	c2:¬EI1 ∨ ¬A ∨ B	cr2: EI1 A B*
R3:Si EI1 et A Alors ¬B	c3:¬EI1 ∨ ¬A ∨ ¬B	cr3: EI1 A ¬B*
R4:Si ¬A Alors C	c4: A ∨ C	cr4: ¬A C*
R5:SI EI3 Alors ¬C	c5: ¬EI3 ∨ ¬C	cr5: EI3 ¬C*

- Vérification par MELOMIDIA

> Elimination des clause-règle annexes

cr2 et cr3 sont éliminées puisque A est non initial et A n'est pas déductible

=> cr2 et cr3 sont indéclenchables

> Coupure sur la variable C. On obtient le système:

cr1 : EI1 EI2 ¬A*

cr45: ¬A EI3

> Coupure sur la variable A. On obtient le système:

cr145: EI1 EI2 EI3

> Arrêt

La base de règles est incohérente. Une BFI qui contient au moins les faits EI1, EI2 et EI3 provoquera la production de contradictions lors de l'inférence .

- Vérification par MELODIA

> Coupure sur la variable B. On obtient le système:

c1 : ¬EI1 ∨ ¬EI2 ∨ ¬A

c23: ¬EI1 ∨ ¬A

c4 : A ∨ C

c5 : ¬EI3 ∨ ¬C

> Coupure sur la variable C. On obtient le système:

c1 : ¬EI1 ∨ ¬EI2 ∨ ¬A

c23: ¬EI1 ∨ ¬A

c45: A ∨ ¬EI3

> Elimination des subsomptions

La clause c1 est subsumée par la clause c23

=> on élimine c1.

Il reste: c23: ¬EI1 ∨ ¬A

c45: A ∨ ¬EI3

> Coupure sur la variable A. On obtient le système:

c2345: ¬EI1 ∨ ¬EI3

> Arrêt ⇒ conjecture d'incohérence: EI1 ∧ EI3

Là encore, l'algorithme ne donne pas les bonnes BFI à interdire à l'entrée du moteur. Si une BFi contient seulement EI1 et EI3, elle ne permet pas la déduction de faits contradictoires lors de l'inférence.

c) Exemple pratique

Nous allons ici fournir un exemple "concret" de la différence des deux démarches. Il est un peu long, aussi nous ne ferons que survoler les étapes suivies par les deux vérificateurs.

c.1) La Base de Règles

Soit la base de règles:

```
REGLE UN-UN
SI   A = 1          ALORS     B = 'vrai'

REGLE UN-DEUX
SI   A = 2          ALORS     B = 'faux'

REGLE UN-TROIS
SI   B = 'vrai'          ALORS     E = 1

REGLE UN-QUATRE
SI   B ≠ 'vrai'          ALORS     E = 2
;-------------------------------------------------;
REGLE DEUX-UN
SI   A = 3          ALORS     C = 'vrai'

REGLE DEUX-DEUX
SI   A = 4          ALORS     C = 'faux'

REGLE DEUX-TROIS
SI   C = 'vrai'          ALORS     E = 2

REGLE DEUX-QUATRE
SI   C ≠ 'vrai'          ALORS     E = 3
;-------------------------------------------------;
REGLE TROIS-UN
SI   A = 5          ALORS     D = 'vrai'

REGLE TROIS-DEUX
SI   A = 6          ALORS     D = 'faux'
```

REGLE TROIS-TROIS
SI D = 'vrai' ALORS E = 1

REGLE TROIS-QUATRE
SI D ≠ 'vrai' ALORS E = 3

c.2) Comportement de MELODIA

Cette base de règles est transformée par MELODIA en un ensemble de clauses.
On note a_i le nombre représentant l'expression A = i,
e_i le nombre représentant l'expression E = i,
bv et bf les nombres représentant respectivement (B = 'vrai') et (B = 'faux'),
cv et cf les nombres représentant respectivement (C = 'vrai') et (C = 'faux'),
dv et df les nombres représentant respectivement (D = 'vrai') et (D = 'faux').

En raison de la symétrie évidente de la BR, on présente l'ensemble des clauses sur 3 colonnes correspondant aux trois blocs constitutifs de la BR.

cl1:	-a1	v	bv	cl5:	-a3	v	cv	cl9:	-a5	v	dv
cl2:	-a2	v	bf	cl6:	-a4	v	cf	cl10:	-a6	v	df
cl3:	-bv	v	e1	cl7:	-cv	v	e2	cl11:	-dv	v	e1
cl4:	bv	v	e2	cl8:	cv	v	e3	cl12:	dv	v	e3

A cet ensemble s'ajoutent:
* les 3 clauses d'exclusion sur les 3 variables représentant l'entité E:

clxE1: -e1 v -e2
clxE2: -e2 v -e3
clxE3: -e1 v -e3

* les 3 clauses d'exclusion sur les couples (bv, bf), (cv, cf) et (dv, df):

clxB: -bv v -bf
clxC: -cv v -cf
clxD: -dv v -df

* les 15 clauses d'exclusion entre les 6 variables représentant l'entité A qui sont toutes de la forme

clxAj: $-a_i$ v $-a_k$ avec i <k et i et j compris entre 1 et 6.

Le vérificateur de la BC-Cohérence de MELODIA répond pour ce système qu'il est insatisfiable. En effet, on a le phénomène suivant:

- le couple de clauses: -bv v e1 , bv v e2
donne, par coupure sur bv, la clause: e1 v e2
- le couple de clauses: -cv v e2 , cv v e3
donne, par coupure sur cv, la clause: e2 v e3
- le couple de clauses: -dv v e1 , dv v e3
donne, par coupure sur dv, la clause: e1 v e3

On se retrouve donc après les coupures sur bv, cv et dv avec dans le système les six disjonctions

cl3-4:	e1	v	e2	
cl7-8:	e2	v	e3	Clauses obtenues par coupure
cl11-12:	e1	v	e3	
clxE1:	-e1	v	-e2	
clxE2:	-e2	v	-e3	Clauses d'exclusion associées à E
clxE3:	-e1	v	-e3	

qui est un système de clauses clairement insatisfiable.

Cette erreur vient du fait que, dans la logique de MELODIA, soit B (ou C ou D ou E) vaut 'vrai', soit il vaut quelque chose de différent de 'vrai'. Il n'est pris en considération le fait qu'il puisse ne pas exister.
Donc des deux clauses non (B = vrai) OU (E = 1) et (B = vrai) OU (E = 2)
il déduit (E = 1) OU (E = 2).

On a donc, d'après MELODIA, une base de règles totalement contradictoire. Donc toute base de faits initiale valide doit déclencher des démonstrations contradictoires!

c.3) Comportement de MELOMIDIA

Le vérificateur de MELOMIDIA est plus pragmatique. On sait que la traduction est identique à celle ci-dessus aux nuances suivantes près:

- le littéral représentant le conséquent est étoilé,
- nous n'inversons pas le signe des littéraux dans nos clauses, et ne les considérons donc pas comme des disjonctions mais comme des productions. On obtient donc les clauses-règles suivantes:

cl1:	a1	bv*	cl5:	a3	cv*	cl9:	a5	dv*
cl2:	a2	bf*	cl6:	a4	cf*	cl10:	a6	df*
cl3:	bv	e1*	cl7:	cv	e2*	cl11:	dv	e1*
cl4:	-bv	e2*	cl8:	-cv	e3*	cl12:	-dv	e3*

- Par ailleurs, nos clauses d'exclusion sont:
  * les 3 clauses d'exclusion sur les 3 variables représentant l'entité E:

clxE1:	e1	e2
clxE2:	e2	e3
clxE3:	e1	e3

  * les 3 clauses d'exclusion sur les couples (bv, bf), (cv, cf) et (dv, df):

clxB:	bf	-bv*
clxC:	cf	-cv*
clxD:	df	-dv*

  * les 15 clauses d'exclusion entre les 6 variables représentant l'entité A qui sont toutes de la forme: clxAj: a_i a_k avec $1 \leq i < k \leq 6$

MELOMIDIA opére d'abord les coupures sur bv et bf, cv et cf, dv et df, qui conduisent au système (on ne répète pas les clauses d'exclusion):

cl1-3:	a1	e1*	cl5-7:	a3	e2*	cl9-11:	a5	e1*
cl2-x-4:	a2	e2*	cl6-x-8:	a4	e3*	cl10-x-12:	a6	e3*

Puis, il opère les coupures sur e1, e2 et e3. A chaque fois la clause obtenue est subsumée par une clause d'exclusion:

- coupure sur e1. On obtient les nouvelles clauses:

cl1-3-x1: a1 -e2
cl1-3-x2: a1 -e3
cl9-11-x1: a5 -e2
cl9-11-x2: a5 -e3

- coupure sur e2. On obtient (partiel) les nouvelles clauses:

cl[1-3-x1]-[5-7]: a1 a3
cl[9-11-x1]-[5-7]: a5 a3

qui sont subsumées par les clauses d'exclusion associées à A.

Il ne nous reste donc à la fin que les quinze clauses d'exclusion associées à l'entité A.

Celles-ci constituent les conjectures d'incohérence du SF (cf Théorème 3 du chapitre III). Mais comme elles correspondent aux contraintes naturelles sur les faits (par exemple, la clause d'exclusion a1 a3 signifie que si on a le fait A = 1, on ne peut avoir en même temps le fait A = 3), on ne les considère pas comme telles.

MELOMIDIA déclarera donc la base cohérente.

On vérifiera aisément qu'aucune base de faits initiale valide ne permet d'inférer des faits contradictoires.

PARTIE 2 : Cohérence de Bases de Règles d'Ordre Un

CHAPITRE V: Vue Générale, Vérifications sans Métaconnaissance

1) Introduction

Nous avons fait part dans le chapitre 1 de nos intentions concernant la vérification de bases de connaissances d'ordre Un. Elles sont ambitieuses et variées. Nous allons dans ce chapitre "planter le décor" et "présenter le prologue" de leur réalisation.

Planter le décor de notre travail signifie répondre aux trois questions suivantes:

1°) sur quoi travaillons-nous ? C'est-à-dire quels sont nos systèmes-cibles, ou, plus précisément, comment sont-ils écrits ? On décrira, dans une première partie, la syntaxe des bases de règles sur lesquelles nous allons travailler, en insistant sur la spécificité de cette syntaxe par rapport aux langages d'ordre un "usuels".

2°) avec quoi travaillons-nous ? C'est-à-dire quel est notre outil d'investigation de ses bases de règles ? En concevant notre vérificateur sous la forme d'un système expert, nous avons donné une réponse originale à cette question. Nous justifierons ce choix et décrirons l'interface nécessaire entre les bases de règles cibles et les bases de règles vérificatrices.

3°) comment travaillons-nous ? C'est-à-dire dans quel "style" allons nous procéder à ces vérifications ? On exposera brièvement, dans la partie "Postulats Méthodologiques", les grandes orientations qui conduisent nos traitements.

Ces trois points occuperont les parties 2) et 3) de ce chapitre.

Dans la tragédie antique, le choeur a pour rôle (entr'autre) de commenter l'action et de révéler au public, en les amplifiant, les caractères des protagonistes. Le prologue est la partie de la pièce qui se déroule avant l'entrée en scène du choeur.

Nous présenterons au chapitre suivant ce qui constituera notre "choeur". Nous le verrons être constitué d'un ensemble de métaconnaissances, regroupées dans la base de métafaits. Ici, nous donnons le prologue. Nous allons en effet décrire tout ce que notre vérificateur fait avant que n'intervienne la base de métaconnaissances. Il s'agit, comme au théâtre, à la fois d'un préliminaire court et sans influence fondamentale sur la suite, et d'un ensemble d'actions très significatives de l'état d'esprit qui conduira les actions futures.

Plus concrètement, on décrira ici de vérifications heuristiques tendant à détecter les erreurs lexicales présentes dans la base de règles cible. On commencera par donner le principe ou l'heuristique que l'on cherche à vérifier. Puis sera fournie la description de la méthode pour y parvenir.

2) Description du langage d'entrée de VAUBAN1

La présentation qui va être faite ici est basée sur les notices de description des langages BOOJUM et GENESIA2. On se reportera à [DORMOY86a] , [DORMOY86b] et [STERIA 87] pour une description plus détaillée de ces deux générateurs de systèmes experts, et à l'annexe 2 pour certaines précisions sur le langage. Certains points et exemples sont tirées de [DORMOY89].

La syntaxe des bases de règles vérifiée par VAUBAN1 est un sous-ensemble de la syntaxe de GENESIA2. Celui-ci est un langage à base de règles d'ordres 0, 1 et 2, dont le moteur d'inférence associé fonctionne en chaînage avant. Il contient la possibilité d'avoir des prémisses entièrement variables (ordre 2) et d'exprimer des faits qui parlent de faits.

Nous présenterons d'abord la syntaxe des faits, ce qui nous permettra d'aborder commodément celle des règles.

a) Les faits

Un fait, comme en logique Zéro Plus, est un doublet :

<Fait> ::= <Objet> ' <Objet>

Le second objet est la valeur, que l'on appelle le **statut**, du premier.
Dans GENESIA2, **chaque objet possède à tout instant un statut unique.**

Le concept d'objet est défini, en terme de grammaire BNF de la manière suivante:

<Objet> ::= <Objet_Atomique> / (<Objet> <Objet> <Objet>)
avec <Objet_Atomique> ::= <un mot non entouré de parenthèses> / <Nombre>

C'est-à-dire, moins formellement:

1) une constante ou un nombre est un objet
2) un triplet ordonné d'objets est un objet

Voici quelques exemples de faits:

TOTO Père LULU ' VRAI
ZONZON ' MENTEUR
ZONZON Sait (TOTO Père LULU) ' (A_VERIFIER Dans LISTE_SAIT)
(((P Sait A) Et (P Sait (A ==> B))) --> (P Sait B))

Notons que cette syntaxe repose donc sur la notion de **relation binaire** (Père, Sait, Et, --> dans l'exemple ci-dessus), mais permet **l'imbrication** à un degré quelconque des relations les unes dans les autres. On parle alors de triplets emboités.

Précisons:

. que le statut VRAI est affecté par défaut à un fait écrit sans statut. On pourra donc l'omettre et écrire par exemple le premier fait de l'exemple: TOTO Père LULU.

. que la constante INEXISTANT, quand elle est placée en place Statut, signifie pour le moteur que le fait concerné ne doit plus être considéré comme faisant partie de la base de faits. On "tue" donc un fait en lui affectant ce statut.

b) Les règles

Nous dressons ici un panorama de la syntaxe traitée par VAUBAN1. Nous ne cherchons donc pas à décrire exhaustivement le langage GENESIA2.

Une base de règles admissible en entrée de VAUBAN1 est un ensemble ordonné d'expressions de la forme:
REGLE <nom de la règle >
SI[1] <Partie-Prémisse>
ALORS <Partie-Conséquent>

dans laquelle <Partie-Prémisse> est un ensemble de prémisses et <Partie-Conséquent> est un ensemble de conséquents.

+ On va distinguer deux types de prémisses. Celui emprunté à la syntaxe des termes d'un langage d'ordre un (le plus riche), et celui permettant l'expression de comparaisons entre des grandeurs.

> On a donc le premier type:

<Prop-type1> ::= <Terme1> <Objet_Atomique> <Terme3> { ' <TermeS>} { : <Variable>}

* dans lequel <Terme1>, <Terme3> et <TermeS> sont des <Terme>. Un <Terme> est l'équivalent dans les règles d'un <objet> dans les faits. C'est à dire:

- **<Terme>** ::= <Terme_Atomique> /(<Terme> <Terme> <Terme>) avec **<Terme_Atomique>** ::= <Variable> / <Objet_Atomique>

Une Variable est un identificateur entouré de parenthèses. (ex : (X)))

* <TermeS> est appelé le statut de la prémisse. Il n'est pas obligatoire de le mentionner. La prémisse sera alors considérée comme ayant le statut VRAI.
Le " : <Variable>" sert à désigner la prémisse. Il permet d'écrire des prémisses internes à d'autres prémisses, et cela avec le degré d'imbrication que l'on veut (voir les "Facilités d'Ecriture" page suivante).

Exemples:
- (X) PERE (Y), qui est instancié par le fait: JEAN PERE LUC
- (((X) COMPARE_A (Y)) PAR (Z)) IMPLIQUE (T) ' REGLE
qui est instancié par le fait:
((e COMPARE_A f) PAR ">") IMPLIQUE ((f COMPARE_A e) PAR "≤") ' REGLE

---

[1] Une règle de ce type est appelé règle-SI. On peut écrire dans GENESIA2 d'autres type de règles, les règles-TANT_QUE et les règles-QUAND, dont la partie prémisse est introduite respectivement par les mots clés TANT_QUE et QUAND. La différence réside dans le comportement du moteur d'inférence lors de l'examen et plus crucialement lors du réexamen de la règle (voir la notice de GENESIA2 pour plus de détail).
Les règles de ces types sont acceptés par VAUBAN1. Mais elles seront considérées exactement comme des règles-SI, sauf pour la partie 4)b) (nous préciserons alors la nuance introduite).

Facilités d'Ecriture:

1°) On peut écrire une prémisse complexe sous la forme d'une conjonction de plusieurs prémisses en se servant de la possibilité de nommage des prémisses.
Par exemple, on peut écrire la prémisse:

(((X) COMPARE_A (Y)) PAR (Z)) IMPLIQUE (T) ' REGLE

sous la forme:

(X) COMPARE_A (Y) : (EXPR1)
(EXPR1) PAR (Z) : (EXPR2)
(EXPR2) IMPLIQUE (T) ' REGLE

Ces deux formes sont exactement identiques.

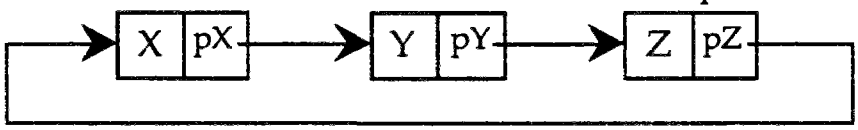
2°) On peut écrire ces prémisses sous forme "fonctionnelle", c'est-à-dire en respectant la règle BNF:

<Prop-type1> ::= <Objet_Atomique><Terme1> = <Terme3>
{ ' <TermeS>} { : <Variable>}

Par exemple, la prémisse: (X) PERE (Y)
peut également s'écrire: PERE(X) = (Y)
Il y a identité sémantique entre les deux formes.

On a donc affaire à un langage puissant permettant l'écriture d'expressions complexes. Par exemple, on peut:

- modéliser simplement des prédicats d'arité quelconques: le prédicat d'arité 5 COURS(MATIERE, SALLE, PROF, JOUR, HORAIRE) peut s'écrire:
COURS ((MATIERE) "," ((SALLE) "," ((PROF) "," ((JOUR) "," (HORAIRE))))) = OUI
- ou décrire des structures de données complexes. Par exemple, la liste circulaire:



sera modélisée par les propositions:

(X) et (PX) : (CELLULE_X)
(PX) POINTE_VERS (CELLULE_Y)
(Y) et (PY) : (CELLULE_Y)
(PY) POINTE_VERS (CELLULE_Z)
(Z) et (PZ) : (CELLULE_Z)
(PZ) POINTE_VERS (CELLULE_X)

On voit ici la richesse qu'apporte le fait de pouvoir dénommer les prémisses.

Bien entendu, cette syntaxe riche nous complique beaucoup la tache. Elle est responsable de la relative simplicité des traitements qui seront proposés ultérieurement.

> Le deuxième type de prémisses regroupe les prémisses des formes:

<Prop-type2> ::= <Terme_Atomique> ' <Comparateur> <Terme_Atomique>
<Prop-type3> ::= <Variable> <Comparateur> <Terme_Atomique>

Un comparateur est une des relations d'ordre usuelles, ie

<Comparateur> ::= < / > / ≥ / ≤ / =

Exemples:

- COMPTEUR ' = 7 qui s'interprète "le statut de COMPTEUR est 7"
- (V)' > (W) qui s'interprète "le statut de l'objet qui instancie la variable (V) est supérieur à l'objet (qui est donc forcément un nombre) qui instancie la variable (W)"
- (V) > (W)
- (V) <= 3

+ Passons maintenant au détail des conséquents:

un **<Conséquent>** est une proposition qui peut prendre 6 formes différentes. Les trois premières sont les équivalents en partie conséquent des trois types de prémisses décrits plus haut, les deux suivantes permettent d'écrire des conséquents "utilitaires" (Entrée/Sortie et création de constantes), et la dernière permet l'écriture de règles à l'intérieur de règles.

Elles sont toutes prises en compte au moins par certains traitements de VAUBAN1, et c'est la raison de leur présence ici. On se reportera à l'annexe 2 pour plus de détails sur certains conséquents.

<Prop-type1'> ::= <Terme1> <Objet_Atomique> <Terme3> { ' <TermeS>}{ : <Variable>}
qu'on peut aussi écrire sous forme "fonctionnelle":

<Objet_Atomique><Terme1><CA><Terme3> { ' <TermeS>}{ : <Variable>}

* <CA> ::= = / <-- .

CA signifie Code d'Affectation. Le premier code est le signe de l'affectation classique. Le deuxième représente l'opération d'affectation avec mise au statut INEXISTANT de tous les autres triplets <Terme1> <Objet_Atomique> <Terme3'> avec <Terme3'> ≠ <Terme3>.

On parle dans le deuxième cas de conséquent écrasant.

* La première forme de <Prop-type1'> est équivalente à la deuxième avec <CA> = "=".

<Prop-type2'> ::= <Terme_Atomique> '= <Calcul sur des Termes Atomiques>
<Prop-type3'> ::= <Variable> = <Calcul sur des Termes Atomiques>

* Exemples:
  - COMPTEUR ' = ((X) + 7) qui s'interprète "le statut de COMPTEUR est égal à la valeur de l'objet qui instancie X plus 7"
  - (V) ' = (W)
  - (V) = (5 + ABS(W))

<Conséquent_CREER> :c'est un générateur automatique de constantes inconnues de la base de faits courante (voir annexe)

<Conséquent_IO> : : il permet, via les ordres LIRE et ECRIRE d'acquérir ou d'écrire des faits dans une règle (voir annexe)

<Conséquent_POUR_TOUT> : (voir aussi l'annexe)

* Il peut être vu comme la possibilité d'écrire une sous-règle dans la partie conséquent d'une règle. Sa syntaxe est:

<Conséquent_POUR_TOUT> :: = POUR_TOUT <Liste de variables>
TEL_QUE <Partie_Prémisse>
DEDUIRE <Partie_Conséquent>

* La liste de variables du POUR_TOUT contient l'ensemble des variables locales à ce POUR_TOUT. Le moteur déclenchera la partie conséquent du POUR_TOUT autant de fois qu'il existe d'instanciations globales différentes de cette liste satisfaisant la partie prémisse du POUR_TOUT, avant de passer au conséquent suivant.

*Exemple: (voir [FALINO89] et [FALINO90]).

```
REGLE Développement_prioritaire_d_un_noeud_terminal
SI    (action) Réalisable_A_Partir_De (situation_c)
      (situation_c) '= COURANTE
      (action) Fixe (etat)
      (situation_f) Est_Défini_Par (etat)
      (situation_f) '= FINAL
ALORS
      (situation_c) Conduit_A CHOIX_PRIORITAIRE
      CREER Situation(X)
      (Y) = CONCATENATION_DE (LIBELLE(situation_c), "-" , (action))
      Libelle(X) = (Y)
      POUR_TOUT ((etat_sur))
      TEL_QUE ((action) Fixe (etat_sur))
      DEDUIRE ((X) Est_Defini_Par (etat_sur))
      POUR_TOUT ((etat_pot))
      TEL_QUE ((etat_pot) Est_Un_Objectif_De (action))
      DEDUIRE ((X) Est_Defini_Par (etat_pot) ' POTENTIEL
```

Remarque:
Le conséquent (Y) = CONCATENATION_DE (LIBELLE(situation_c), "-" , (action)) a pour effet d'instancier la variable (Y) par la concaténation:

- du troisième objet (ou d'un des troisièmes objets) du triplet (situation_c) LIBELLE ... ,
- de "-"
- de l'instanciation de la variable (action).

CONCATENATION_DE est un mot clé du langage GENESIA2.

c) Description sommaire du fonctionnement du moteur d'inférence

On a vu (note lors de la présentation de le syntaxe des règles) que l'on considérerait dans VAUBAN1 toutes les règles comme des règles-SI , sauf exception. Sur ce type de règles, le moteur adopte la stratégie suivante:

a) examen de la base de règles en partant du début, c'est-à-dire de la première règle dans le fichier

b) déclenchement de la première règle possédant une instanciation globale de son ensemble de variables qui n'a pas déjà été utilisé dans un déclenchement précédent de la règle.

c) recherche d'un autre déclenchement de la même règle avec une instanciation globale différente. En cas d'impossibilité, reprise de l'examen des règles à partir de la première d'entr'elles (comme au début de l'inférence).

Autrement dit,

a) GENESIA2 examine une règle après avoir examiné toutes les règles placées avant elle dans la base de règles sans avoir pu en (re-)déclencher une seule.

b) il effectue successivement tous les déclenchements possibles de la règle, avant de chercher une autre règle à examiner.

c) il ne redéclenche pas une règle avec un ensemble d'instanciation de ses variables ayant déjà été utilisé pour un déclenchement antérieur.

3) VAUBAN1: Vue Générale

a) Postulats Méthodologiques

a.1) Un Système Expert

Lorsque nous avons commencé l'étude de la cohérence de bases de règles d'ordre Un, nous voulions éviter d'axer notre travail dans une seule direction. Nous préférions adopter une démarche plus empirique, basée sur l'intuition et les résultats issus des premiers essais. Ainsi avions-nous besoin d'une méthodologie de développement:

- souple: On était sur en effet de devoir revenir souvent sur des parties déjà réalisées, ou de changer souvent d'idée en cours de réalisation.
- modulaire: Nous voulions explorer différents aspects du problème. Nous nous doutions que les programmes correspondants auraient une forte indépendance, tout en devant certainement mettre en commun une partie de leurs déductions.
- facile et rapide à développer: Nous nous voulions opportuniste, prêt à tenter, juste "pour voir", des contrôles d'un nouveau type. D'autre part, nous étions convaincu que seul le test sur des cas concrets de nos idées (issues de l'examen de cas concrets) pourrait nous garder de (trop) s'écarter du pragmatisme de rigueur quand on veut créer un nouveau service (ce qui est la finalité de tout contrôle automatique). Nous voulions donc pouvoir réaliser en quelques jours l'esquisse d'un traitement envisagé, et le tester aussitôt[2].

Dès lors, la méthodologie système expert semblait naturelle, puisque répondant à ces exigences. De plus, en pratique :

- la plupart des concepteurs de systèmes de vérification de base de connaissances se plaignent du manque d'utilisateurs de leur système, particulièrement pendant la phase de réalisation. Les volontaires pour essuyer les plâtres d'un système balbutiant, en effet, ne sont pas légion. Nous avons, d'une certaine manière, résolu ce problème. En effet, nous étions sûr, à tout moment, de pouvoir trouver au moins un système en cours de développement (donc plein d'erreurs potentielles) à tester, et un utilisateur compréhensif: notre système et nous-même! **Notre système étant écrit dans la syntaxe des systèmes qu'il vérifie, il pouvait (toujours au moins) se vérifier lui-même**[3].

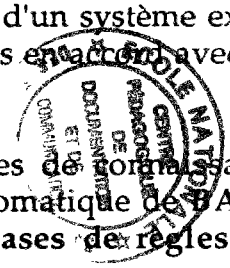
---

[2] Nous ne parlerons dans la suite que des essais fructueux. Nous aurions pu aussi écrire un chapitre sur les "fausses pistes" suivies plus ou moins longtemps, dans lequel aurait pris place une partie sur notre tentative de transformation des BdR en formule d'une algèbre "d'ordre zéro", comme celle développée dans [CASEAU87], et qui s'avéra inadapté à notre contexte; une partie sur notre essai d'utilisation massive des processus de compréhension du langage naturel; une autre sur la tentative d'abstraction de règles (écriture automatique de la spécification correspondant à un paquet donné de règles); ou encore sur notre tentative d'étude de la non-commutativité par rapport à la base de faits initiale ...

[3] Nous eûmes ainsi la délicieuse surprise de voir notre système se trouver lui-même quelques vraies erreurs ... et ne pas en voir des dizaines d'autres, ce qui nous indiquait crûment par où continuer le développement!

- sur le plan technique, les traitements de ce genre sont clairement de nature symbolique. Il s'agit en effet, peu ou prou, d'analyser des implications logiques. Ce travail nécessite donc l'emploi d'un langage de manipulation de symboles. On a donc finalement le choix entre un LISP, un PROLOG, un L.O.O. ou le langage d'un générateur de système expert. Mon expérience personnelle, ainsi que la culture du milieu dans lequel j'effectuais ce travail, m'orientaient nettement vers cette dernière solution.

- enfin, nous n'avions pas l'obligation de faire un produit, mais seulement des maquettes. De plus, nous avions de grosses machines à notre disposition (IBM 3090) et un langage (GENESIA2, issu de BOOJUM) reconnu très performant (cf [DORMOY88]). La faisabilité d'un système expert de vérification était donc a peu près garantie, et ses performances en accord avec nos exigences.



Notre système de vérification de bases de connaissances d'ordre 1, baptisé VAUBAN1 (acronyme de Vérification AUtomatique de BAses de règles de Niveau 1) est donc constitué **d'un ensemble de bases de règles** écrites dans la syntaxe **GENESIA2.**

Partant de ce principe, nous avons développé une interface entre les bases de règles cible et notre système expert. Cette interface consiste naturellement en la **traduction en faits GENESIA2 des règles GENESIA2 à vérifier.**

Notre système, **VAUBAN1,** est donc un système expert. Il est écrit en GENESIA2, et analyse la traduction sous forme de faits de bases de règles GENESIA2

a.2) Autres Postulats Méthodologiques

Notre postulat de base était donc d'écrire un système expert. Les autres grands principes que nous avons respectés sont:

- En premier lieu, de concevoir le processus de vérification comme **un service "à la carte"** offert à l'utilisateur: ce sont en effet les connaissances fournies par l'utilisateur sur son application qui déterminent, à quelques exceptions près, les vérifications opérées par VAUBAN1. Ces connaissances sur une autre connaissance (la base de règles) sont appelées **Métaconnaissances.**

Donc, si l'utilisateur fournit peu de métaconnaissances, VAUBAN1 ne fait (presque) rien. Par contre, s'il fournit davantage de métaconnaissances, VAUBAN1 travaillera **plus,** et **mieux.**

VAUBAN1 n'opère donc pas une vérification de la base de règles par rapport à un modèle statique, mais une vérification de l'adéquation de la base de règles avec d'autres connaissances la concernant. Ces autres connaissances, **les métaconnaissances,** sont fournies par l'utilisateur.

Enfin, cette métaconnaissance est écrite sous forme de faits GENESIA2. On appelle ces derniers les **métafaits,** et leur ensemble **la métabase de faits (MBF).**

- le deuxième principe est d'**avoir des traitements les plus locaux possibles.** C'est-à-dire de traiter un problème au niveau auquel il se pose. Par exemple, si le problème concerne une règle, c'est l'examen de la règle qui tranchera la contradiction potentielle. Ce caractère "local" de l'examen conduit systématiquement à ne définir "que" des **conditions suffisantes** de validité. Par contre, ce type de contrôle est facile à mettre en oeuvre et peu couteux en temps CPU. Cela nous a paru préférable à des contrôles de type "CNS de validité" plus subtils mais exclusifs (car trop couteux en ressources).

- Réciproquement, de même que nous venons de dire que nous vérifierons mal des vrais problèmes, **nous acceptons aussi de vérifier bien des problèmes douteux.** En effet, notre modèle laisse une large place à l'heuristique. On cherchera vainement ici un modèle logique parfaitement rigoureux. Si chaque opération est menée avec un maximum de rigueur, l'idée conductrice est de ne pas refuser de se tromper, tant que l'utilisateur peut lui-même trancher s'il y a erreur ou pas. Nous fournirons tant le critère vérifié que le "coupable" présumé, le critère pouvant lui-même parfois porter à discussion.

- enfin, nous essayerons de vérifier une base de connaissances à **différents niveaux**, tant "**microscopiques**" que "**macroscopiques**". Nous voulons en effet effectuer des vérifications en partant du niveau élémentaire (la proposition, ie une prémisse ou conséquent), pour ensuite s'intéresser aux paires de propositions (liées par une variable), puis à toute une partie (prémisse ou conséquent) de règle, puis à toute la règle, puis aux paires de règles, au paquet de règles, pour arriver enfin à la base dans son intégralité. Loin de nous bien sur la prétention d'affirmer que tout est totalement vérifié et contrôlé! Nous voulons seulement voir quels sont les problèmes qui se posent à chaque niveau, et tenter de les résoudre.

b) Synoptique de VAUBAN1

On va terminer cette présentation par un survol de l'ensemble des modules constituant VAUBAN1. Ces modules se repartissent sur deux couches:

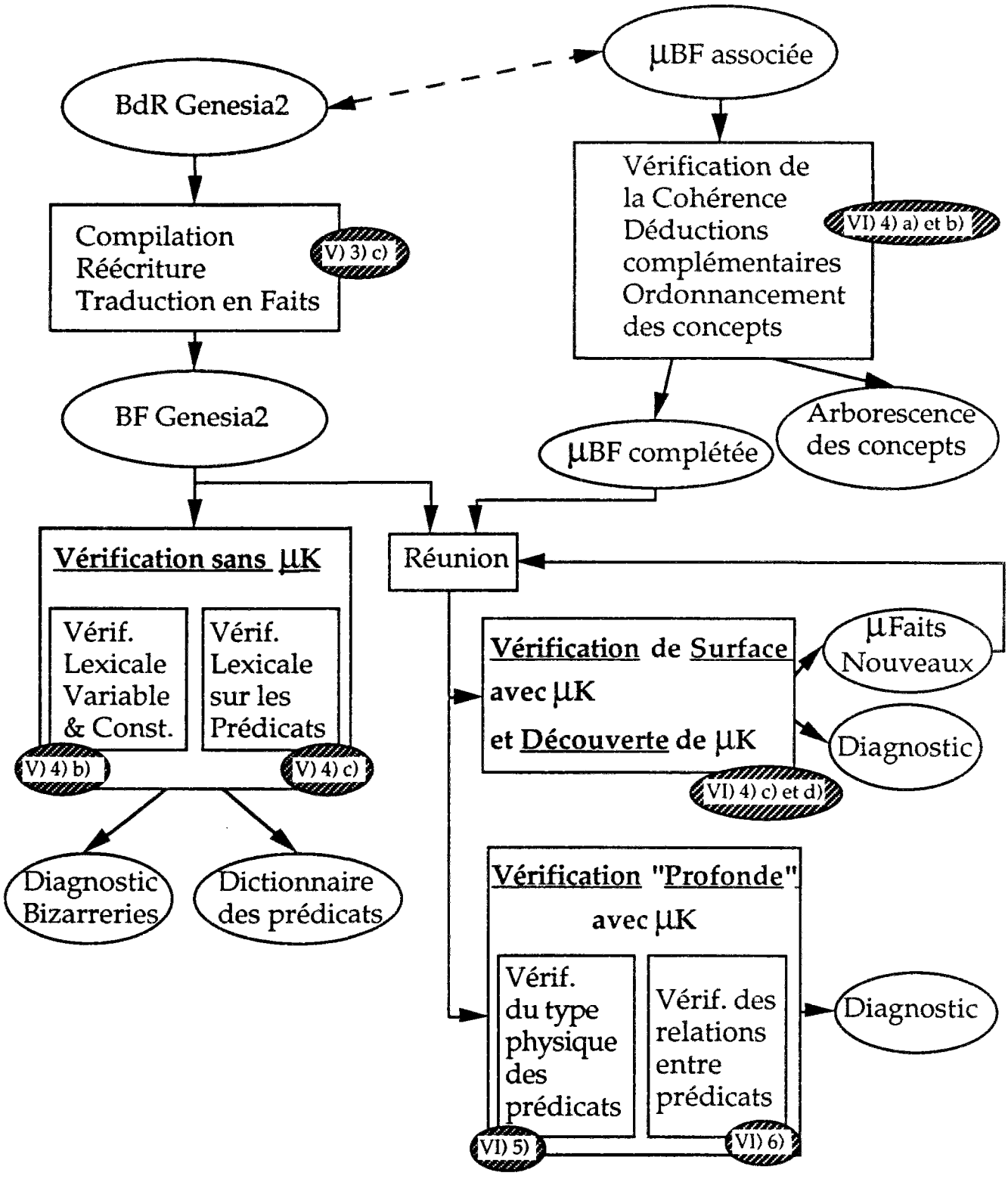
A partir des "entrées", la Base de Règles (notée **BdR** en abrégé) et la MBF, on procède à des opérations de mise en forme: Traduction pour la BdR, vérification de la cohérence et complémentation pour la MBF. Puis, à l'issue de ces premiers traitements, on procède à des vérifications se répartissant en deux catégories:

- les vérifications sans Métaconnaissance, ie par rapport à un modèle fixe,
- les vérifications avec la Métaconnaissance, qui se divisent elles-mêmes en deux sous-catégories:
  * les vérifications utilisant les métaconnaissances "de base", ces vérifications allant ici de pair avec la déduction de nouvelles métaconnaissances,
  * les vérifications grâce aux métaconnaissances plus "profondes".

Dans le diagramme de la page suivante, on utilise les notations suivantes:
μ = méta, K = connaissance(s), Const. = Constante (= objet)
BdR, BF, μBF = Base de Règles, Base de Faits, MétaBase de Faits

Les carrés représentent les traitements, les ovales sur fond clair les fichiers, les ovales sombres contiennent la référence du paragraphe de description du traitement sur lequel ils sont posés:

SYNOPTIQUE de VAUBAN1



Nous allons maintenant donner une description sommaire du premier maillon de la chaîne de traitements d'une base de règles, à savoir les opérations de réécriture et de traduction en faits GENESIA2.

c) le Traducteur de bases de connaissances en faits

Les paragraphes de cette partie décrivent la facette "programmation classique" de notre travail. En effet, le traducteur est un programme Pascal, dérivé du du moteur d'inférences de GENESIA2 et inspiré du traducteur pour BOOJUM de Jean-Luc DORMOY (évoqué dans [DORMOY90]). Il est néanmoins le résultat d'un effort important de développement, parce que nous voulions qu'il puisse traiter toutes les combinaisons de la syntaxe GENESIA2. Nous voulions en effet pouvoir traduire des "vraies" bases de règles (par exemple VAUBAN1 lui-même), tant pour notre propre usage que dans la perspective (à l'étude) d'autres travaux sur ces bases de règles dans notre groupe de travail (par exemple, la compilation de BdR, ou l'explication d'un échec à l'inférence à partir des résultats de cette inférence ...).

c.1) Les opérations de Réécriture sur la Base de Règles

Nous avons dans la partie 1) donné la syntaxe de GENESIA2. Elle permet d'écrire des formules condensées qui, laissées ainsi, multiplieraient les règles de traitement. On va donc décrire maintenant trois procédés de normalisation et de simplification des phrases de ce langage.

c.1.1) Le Déboîtement

On a vu qu'on pouvait écrire des prémisses ou des conséquents du type
<Terme1> <Objet_Atomique> <Terme3> { ' <TermeS>} { : <Variable>}
avec <Terme1>, <Terme3> et <TermeS> sont des Termes.
Par exemple, on peut écrire:

- exemple1: (X) Sait ((Y) Père LULU) ' FAUX
- exemple2: (((E1) COMPAREE_A (F1)) PAR (REL1))
  EQUIVAUT_A (((E2) COMPAREE_A (F2)) PAR (REL2)) ' (S)

Nous allons introduire de nouvelles variables et casser ces propositions. Les deux propositions avant et après transformation sont rigoureusement identiques pour le moteur d'inférences. Cette transformation permet d'unifier le nombre de termes présents dans la prémisse.
Sur les deux exemples ci-dessus, le résultat de cette opération sera:

- exemple1: (Y) Père LULU : (var$1)
  (X) Sait (var$1) ' FAUX
- exemple2: (((E1) COMPAREE_A (F1)) : (var$1)
  (var$1) PAR (REL1)) : (var$2)
  (((E2) COMPAREE_A (F2)) : (var$3)
  (var$3) PAR (REL2)) : (var$4)
  (var$2) EQUIVAUT_A (var$4) ' (S)

(var$1), (var$2), (var$3), (var$4) sont les nouvelles variables introduites.

A l'issu de la transformation, on se retrouve avec une série de prémisses du type:
<Terme_Atomique1> <Objet_Atomique> <Terme_Atomique3>
{ ' <Terme_AtomiqueS>} { : <Variable>}
c'est-à-dire ne contenant plus que des termes atomiques.

c.1.2) Formulation explicite des statuts

Dans le même ordre d'idée, la syntaxe permet d'écrire des prémisses du type
<Prop-type2> ::= <Terme_Atomique> ' <Comparateur> <Terme_Atomique>
avec <Comparateur> ::= < / > / ≥ / ≤ / =
ou des conséquents du type
<Prop-type2'> ::= <Terme_Atomique> '= <Calcul sur des Termes Atomiques>
Par exemple, on peut écrire:

- exemple1: COMPTEUR '≤ 7
- exemple2: (V)' > (W)
- exemple3: (V)' = (5 + Abs(W))

Pour diminuer la complexité des prémisses, on va réécrire les prémisses de ce type en deux prémisses plus élémentaires, l'une servant à expliciter le statut du terme gauche, et l'autre comparant ce statut au terme droit.

Par exemple, les prémisses de l'exemple précédent s'écriront:

- exemple1: COMPTEUR '= (var$1)
  (var$1) ≤ 7
- exemple2: (V) ' = (var$1)
  (var$1) > (W)
- exemple3: (V) ' = (var$1)
  (var$1) = (5 + Abs(W))

On aura besoin au paragraphe suivant de désigner les propositions de la forme
<Terme_Atomique> '= <Variables Générées>
(par exemple COMPTEUR '= (var$1))
On dira donc que ces propositions sont du type "**<Prop-type-statut-élémentaire>**"

Remarque:

Cette transformation et la précédente résultent directement du mode de stockage des prémisses de l'analyseur syntaxique de GENESIA2. Nous nous sommes donc contenté de traduire dans la représentation en faits la représentation de ces propositions dans le moteur d'inférences.

c.1.3) Introduction de la fonction QUOTE

Enfin, on sait que chaque objet GENESIA2 possède un statut, qui est par défaut l'objet atomique VRAI . On considérera dorénavant le statut d'un objet comme l'image de cet objet par la relation **QUOTE**.

Cette transformation est presque totalement légitime. En effet le statut d'un objet n'est en effet que l'image de cet objet par une fonction, c'est-à-dire par une relation monovaluée. Il n'est donc pas choquant d'expliciter cette relation, d'autant plus que nous verrons que nous aurons le moyen d'exprimer dans notre modèle le caractère monovalué de cette relation.

Toutefois, d'une part, le statut INEXISTANT est une valeur à laquelle correspond un comportement particulier du moteur, ce qui sort de notre modèle. Nous essayerons néanmoins d'en tenir compte (voir chapitre 6, 4) a) a.2)). Mais surtout, la contrainte de monotonie, pilier de la partie 4) du chapitre 6 et la notion de domaine conceptuel (voir 2) du chapitre 6) conviennent assez mal ici.

Concrètement, les propositions du type <Prop-type1>, <Prop-type1'> et <Prop-type-statut-élémentaire> seront "réécrites" en introduisant la fonction QUOTE, et on explicitera le statut VRAI par défaut des prémisses dont le statut n'est pas mentionné :
Par exemple, les prémisses:
- (V) ' = (var$5)
- F(x) = y
- (X) Sait (var$8) ' FAUX

deviendront respectivement:
- QUOTE (V) = (var$5)
- F(x) = (y) : (var$1)
  QUOTE (var$1) = VRAI
- (X) Sait (var$8) : (var$9)
  QUOTE (var$9) = FAUX

La relation QUOTE[4] est donc ainsi rajoutée à tout système en entrée de VAUBAN1.

A l'issue de ces transformations, il ne reste que les prémisses de type <Prop-type1> simple (sans triplets emboités) et <Prop-type3>. De même, il ne reste que des conséquents de type <Prop-type1'> simple, <Prop-type3'> et les conséquents "spéciaux" (CREER, IO, POUR_TOUT).
On peut alors aborder la phase de traduction proprement dite.

---

[4] On donne ici, par anticipation, les métafaits associés à QUOTE:
Quote DomaineConceptuelObjet ObjetGenesia
Quote RecouvrementObjet TOTAL
Quote DomaineConceptuelValeur ObjetGenesia
Quote RecouvrementValeur PARTIEL
Quote TypePhysique FONCTION
Quote TypeFonctionnel ATTRIBUT
Le concept ObjetGenesia est déclaré comme incluant tous les autres concepts.

c.2) Traduction des Bases de Règles en Bases de Faits

La traduction est la plus simple possible, sachant qu'ensuite on veut pouvoir avoir accès à chaque partie d'un triplet de façon indépendante et sans avoir à écrire dans notre système de prémisses entièrement variable (ie de la forme (F) (X) = (Y)).

Plutôt que de donner une spécification rigoureuse, on va seulement donner le résultat de la traduction sur quelques propositions.

- Les propositions: (C) Possede (S) ' (NBMOIS)
  (NBMOIS) > 6

Si elle sont respectivement les première et deuxième prémisses de la troisième règle de la base seront écrites:

```
(SI3$1 Premisse R3) PositionPremisse "SI"
SI3$1 Element Premisse
SI3$1 Terme1PCTriplet (C Element Variable)
SI3$1 Terme2PCTriplet (Possede Element Constante)
SI3$1 Terme3PCTriplet (S Element Variable)
SI3$1 ObjetPCTriplet (var$1 Element Variable)

(SI3$2 Premisse R3) PositionPremisse "SI"
SI3$2 Element Premisse
SI3$2 Terme1PCTriplet (var$1 Element Variable)
SI3$2 Terme2PCTriplet (QUOTE Element Constante)
SI3$2 Terme3PCTriplet (NBMOIS Element Variable)

(SI31$3 Premisse R3) PositionPremisse "SI"
SI3$3 Element Premisse
SI3$3 TermeGauche (NBMOIS Element Variable)
SI3$3 TermeDroit (6 Element Constante)
SI3$3 CodeComparateur ">"
```

- Le conséquent: POUR_TOUT ((PREV))
  TEL-QUE(PREVISION(S) = (PREV))
  DEDUIRE(PREVISION(S) <-- ((PREV)/ 12))

Si il est le troisième de la deuxième règle s'écrit:

```
(ALORS2$3 Consequent R2) NumeroConsequentCR   3
ALORS2$3 Element Consequent
PREV ApparaitVRPourTout   ALORS2$3    (* variables locales du Pour_Tout*)
var$1 ApparaitVRPourTout  ALORS2$3
var$2 ApparaitVRPourTout  ALORS2$3

(TEL_QUE2$3$1 Premisse ALORS2$3) PositionPremisse "TEL_QUE"
TEL_QUE2$3$1 Element Premisse
TEL_QUE2$3$1 Terme1PCTriplet (S Element Variable)
TEL_QUE2$3$1 Terme2PCTriplet (PREVISION Element Constante)
(* etc ...       ==>    On passe à la partie Déduire du Pour_Tout          *)

(DEDUIRE2$3$1 Consequent ALORS2$3) NumeroConsequentCR   1
DEDUIRE2$3$1  Element Consequent
DEDUIRE2$3$1  VariableCalcul var$1
DEDUIRE2$3$1  Calcul ("/" OperantSur ((PREV Element Variable)
CoupleDArguments  (12 Element Constante) ))
```

```
(DEDUIRE2$3$2 Consequent ALORS2$3) NumeroConsequentCR    2
DEDUIRE2$3$2  Element Consequent
DEDUIRE2$3$2   CodeDAffectation "<--"
DEDUIRE2$3$2 Terme1PCTriplet (S Element Variable)
DEDUIRE2$3$2 Terme2PCTriplet (PREVISION Element Constante)
DEDUIRE2$3$2 Terme3PCTriplet (var$1 Element Variable)
DEDUIRE2$3$2 ObjetPCTriplet (var$2 Element Variable)

(DEDUIRE2$3$3 Consequent ALORS2$3) NumeroConsequentCR    3
DEDUIRE2$3$3  Element Consequent
DEDUIRE2$3$3   CodeDAffectation "<=="
DEDUIRE2$3$3 Terme1PCTriplet (var$2 Element Variable)
DEDUIRE2$3$3 Terme2PCTriplet (QUOTE Element Constante)
DEDUIRE2$3$3 Terme3PCTriplet (VRAI Element Constante)
```

4) Vérifications sans Métaconnaissance

a) Introduction

Notre souci majeur est, nous l'avons dit, de traiter sous des angles variés le problème de la vérification de bases de règles d'ordre un. Celui qui va être présenté ici n'est certes pas le plus riche conceptuellement. Il résulte directement de notre choix de réaliser notre outil dans le même langage que celui des bases de connaissances que nous devions vérifier.

Ainsi, alors que nous réalisions la partie de notre système dédiée à des vérifications basées sur la confrontation base de règles / métaconnaissance (cf chapitre 6), nous nous rendîmes compte qu'une bonne partie des erreurs que nous commettions provenaient de "fausses manoeuvres" lors des innombrables opérations de duplication de règle[5].

Les erreurs que nous traquerons ici sont donc plutôt stupides, et les méthodes de détection basées sur les anomalies ou les singularités que ce genre de bévue entraîne. On s'occupe donc ici de la correction lexicale de la base, en vérifiant que la base de règles seule respecte un certain nombre d'heuristiques.

Développons les deux derniers points.

Tout d'abord, le fait de travailler avec la base de règles uniquement est important:
. il montre d'une part que l'on peut aussi effectuer des vérifications (modestes) sans métaconnaissance ou autre C-Modèle (au sens défini dans [AYEL&R90]).

---

5) Prenons un exemple. Tout concepteur de base de règles doit souvent implémenter l'équivalent d'un "case of" informatique, ie construire un ensemble de règles presque identiques à une prémisse et un ou plusieurs conséquents près. Pour réaliser cette implémentation, sa tendance naturelle d'informaticien (c'est-à-dire la flemme, ou, si l'on préfère, le goût de l'utilisation maximale des facilités offertes par la machine), va le pousser à utiliser le "copier/coller" de l'éditeur utilisé. Gare alors aux fautes de frappe - ou de jugeote - que la rapidité de ce type de développement suscite !

. cela aide à convaincre les utilisateurs sceptiques de l'intérêt d'un outil de vérification (ils n'ont, pour utiliser ce module de vérification, rien à faire, si ce n'est attendre les résultats de l'exécution), et les encourage à fournir, pour des traitements plus avancés, la métaconnaissance indispensable.

Ensuite, que ce traitement se fasse par la vérification d'heuristiques révèle que ce module contient, en fait, une partie de notre connaissance - par nature subjective - sur la bonne façon de programmer. Autrement dit qu'il traduit nos manies, nos "trucs" de programmation. Nous avons voulu les rendre publiques, croyant pouvoir affirmer qu'elles permettent de mettre en évidence certains points obscurs des bases de règles examinées, mais aussi pour initier un débat autour de la question : "Syntaxiquement, qu'est ce qu'une bonne règle ?".

Telle sont les raisons qui nous ont amenés à aborder le problème de la vérification par cette voie de traverse. La plupart des règles qui vont être décrites ici pourraient trouver leur place dans un éditeur spécialisé pour bases de règles, et s'appliqueraient on-line (cf le système de contrôle de la cohérence statique de SACCO, voir [AYEL87]), les coûts CPU des contrôles évoqués ici étant négligeables.

b) Vérification lexicale sur les variables et les constantes

b.1) Ambiguïté variable constante

Le premier contrôle qu'effectue notre système repose sur l'**heuristique**:
"Dans une même règle, une variable et une constante ne doivent pas avoir le même nom". On se souvient qu'une variable est un identificateur entouré de parenthèses. On interdit donc de trouver le même identificateur, sans parenthèse, dans la même règle.

Ce contrôle vise deux objectifs:

1) Il permet de détecter l'erreur consistant à oublier de mettre des parenthèses autour du nom d'une variable dans certaines de ces occurrences. Cette erreur est fréquente lorsque, pour augmenter la généralité d'une règle, on transforme dans une règle une constante en variable.

Exemple :
Au départ, on a un système dans lequel la seule relation entre grandeurs est SUP. On a écrit:

SI ((A) Comparé_A (B)) Par SUP
((B) Comparé_A (C)) Par SUP
ALORS
((A) Comparé_A (C)) Par SUP

Puis, d'autres relations devenant nécessaires, on écrit la transitivité des relations d'ordre:

SI ((A) Comparé_A (B)) Par (SUP)
((B) Comparé_A (C)) Par (SUP)
ALORS
((A) Comparé_A (C)) Par SUP

(SUP) s'instanciant dans {SUP, SUPEGAL, INF, INFEGAL}
en oubliant la transformation dans le conséquent, d'où erreur.

2) Il interdit les mélanges volontaire de types, du style
SI NATURE(SITUATION) = SITUATION
qui indique que tout objet instanciant la variable (SITUATION) est du type SITUATION. Même s'il est important que le nom de la variable évoque sa sémantique (rien de pire à comprendre qu'une base dans laquelle toutes les variables s'appellent (X), (Y), (Z), ...), il doit, à mon sens, apparaître une différence syntaxique entre le nom du représentant d'une classe et les objets de cette classe, bien que cet abus soit courant en mathématiques.

b.2) Ambiguïté du lien entre les variables d'une règle

Le deuxième contrôle repose sur une heuristique qui mérite d'être plus détaillé. En première approche, elle consiste à dire, en supposant que deux variables sont liées si elles apparaissent toutes les deux dans la même prémisse: **"Le graphe des liens entre variables doit être connexe"**. Autrement dit, toute variable d'une règle doivent avoir un rapport avec les autres variables de la règle.

Par exemple, cette heuristique signale les règles du type:

SI F(X) = (Y)
G(Z) = (T)
ALORS H(X) = a
K(T) = b

dont la sémantique est un peu étrange:
on prend des couples (X, Y) et (Z, T) qui n'ont a priori rien à voir, et on déduit une propriété sur un élément de chacun de ces couples.
De plus, s'il existe 20 triplets du type (X) F (Y) et 50 du type (Z) G (T), on va faire 1000 inférences pour déduire au plus 70 triplets nouveaux. Voilà qui parait bien maladroit ! Que la variable (Z) soit en fait (Y), créant ainsi le lien (X) — (Y)/(Z) — (T), ou qu'il manque une prémisse du style L(Y) = (Z) ne nous surprendrait pas.

Toutefois, les choses ne sont pas toujours aussi simples. D'une part, des variables peuvent n'apparaitre qu'en conséquent, d'autre part, le lien entre variables présentes en partie prémisse peut aussi se situer dans un conséquent. Nous allons donc découper le traitement en deux parties. En premier lieu, on recon-naitra les liens simples, avant de s'occuper des liens complexes mais licites.

b.3) Le Module GRAPHVAR

Après une partie liminaire contrôlant le respect de l'heuristique décrite en b.1), le module GRAPHVAR se compose des deux parties évoquées ci-dessus.

1ère partie: Détection de l'absence de liens simples entre variables d'une même règle
. buts : Contrôle du respect des heuristiques:
h1: **"Toutes les variables apparaissant en partie prémisse d'une règle doivent être reliées entr'elles en partie prémisse de la règle"**
h2: **"Toute variable apparaissant seulement en conséquent doit être reliée aux variables-prémisse de la règle ou à des variables - conséquent déjà reliées".**

Plus précisément, beaucoup de règles utilisent dans leur partie conséquent de nouvelles variables, appelées variables-conséquent, qu'il faut examiner. Ce sont :

- les variables résultant de l'ordre LIRE
- les variables utilisées pour déboîter des conséquents complexes.
- les variables acceuillant le résultat d'opérations (calcul, concaténation etc ...)

Elles doivent bien entendu être liées aux variables définies en prémisses ou à d'autres variables-conséquent déjà reliées.
Plus précisément, pour les deux premières catégories de variables-conséquent, on applique l'heuristique h2 tel quel. Pour la dernière catégorie, on rajoute la contrainte supplémentaire que lien avec les autres variables doit nécessairement s'effectuer grâce à un autre conséquent que celui qui les définit: par exemple, si V est défini par le conséquent (V) = (W) + 1, on ne tient pas compte du lien entre (V) et (W) établi par ce conséquent. Il faudra donc au moins une autre apparition de (V) en conséquent de la règle pour pouvoir le relier aux autres variables de la règle

Enfin, on a vu qu'un conséquent "POUR_TOUT" pouvait utiliser de nouvelles variables, définies dans sa partie prémisse ou dans sa partie conséquent, et qui lui sont propres. L'interprétation de h1 et h2 dans ce cas est la suivante:

1°) Toute variable déclarée en entête d'un conséquent POUR_TOUT doit être reliée dans la partie prémisse de ce POUR_TOUT:

- soit aux variables définies en partie prémisse de la règle,
- soit aux variables définies dans un conséquent situé avant le POUR_TOUT considéré,
- soit à une autre variable locale du (même) POUR_TOUT qui soit elle-même reliée.

2°) Toute variable conséquent d'un POUR_TOUT doit être reliée aux variables définie dans l'entête du POUR_TOUT, ou à des variables - conséquent déjà reliées".

. Méthode:

Pour savoir si toutes les variables d'une règle sont reliées entr'elles, on construit le graphe **non-orienté** des liaisons entre variables. Ce graphe a pour ensemble de sommets l'ensemble des variables de la règle, et repose sur la relation suivante: Une variable (X) est relié à une variable (Y)

* s'il existe une propositions du type:
- F(X) = (Y) , (: liaison etiq[6] - vat[7])
- F(X) = ?Z : (Y) avec ?Z un terme (: liaison etiq- objet[8])

---

[6] Dans un triplet de la forme a R b : c ' d, on dira que:
a est en place étiq (abréviation de étiquette),
R est en place qual (abréviation de qualité),
b est en place vat (abréviation de valeur attributive),
c est en place objet (puisque c'est lui qui représente l'objet-triplet a R b),
d est en place statut
Ces notations sont issues de [DORMOY87]. Les appellations étiquette, qualité et valeur attributive viennent du moteur ALOUETTE ,cf [MULET&B86]

[7] voir ci-dessus

[8] voir ci-dessus

- F(X) = ?Z ' (Y) ou F ?Z = (X) ' (Y) ou (X) = '(Y) avec ?Z un terme (: liaison etiq/vat/objet - Statut[9])
- (X) > (Y), (avec n'importe quel comparateur: liaison comp)
- F(X) = a ou F a = (X) et G(Y) = a ou G a = (Y)
  F(X) = a ou F a = (X) et a '= (Y) ou (Y) '= a
  (X) '= a ou a '= (X) et (Y) '= a ou a '= (Y)

* s'il existe un calcul entre (X) et (Y) (lien calcul)

L'algorithme consiste:
1°) à choisir arbitrairement une variable-prémisse de la règle,
2°) à générer le graphes des liaisons valides entre variables , identique à celui défini ci-dessus mais dont les arcs respectent en plus les heuristiques définies plus haut
3°) à regarder si cette composante contient toutes les variables de la règle:
Le système signale (warning) les variables qui n'ont pu être intégrées dans la composante construite du graphe

Exemple: Supposons qu'on ait la règle:

```
REGLE ventilation_des_ventes_annuelles_d_un_produit
SI   (produit) Volume_Vente_Annuelle (v)
     Prix_Unitaire_Hors_Taxe(produit) = (puht)
     Catégorie(produi) = (cat)
     Taux_TVA(cat) = (taux)
ALORS
     (message) = CONCATENATION_DE ( "Période considérée d'exploitation du
     produit " (produit) " ? : ")
     ECRIRE(TERMINAL, (message))
     LIRE (TERMINAL, (mois1)(mois2))
     (mois1) A (mois2) : (période)

     POUR_TOUT( (nummois1)(nummois2))
     TEL_QUE( (nummois1) Numéro_Du_Mois_De (mois1)
              (nummois2) Numéro_Du_Mois_De (mois2))
     DEDUIRE( (NBMOISPER) = ((nummois2) - ((nummois1) +1))
              (V) = (produit) * (puht) * 12/(NBMOISPER)
              Chiffre_D_Affaire_Hors_Taxe(produit) = (v) : (caht)
              Chiffre_D_Affaire_TTC(produit) = (v)*(1 + ((taux)/100)) : (cattc)
              (caht) Période (période)
              (cattc) Période (période) )
```

Fonctionnement de la règle: voir la note 10

---

[9] voir note précédente

10 Note sur le fonctionnement de la règle (après correction). Admettons que l'on est en entrée:

Ordinateur_Portable Quantité_Vente_Annuelle 1200
Ordinateur_Portable Prix_Unitaire_Hors_Taxe 15 ; en kF bien sur ! ;
Ordinateur_Portable Catégorie Biens_Industriels
Biens_Industriels Taux 17.6

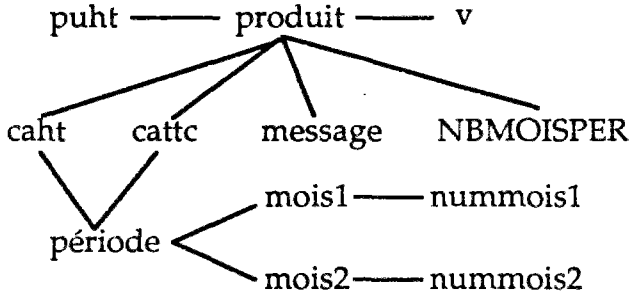
Et qu'on donne comme période: "Janvier Mars"

Déroulement de l'algorithme:
On a quatre ensembles de variables (on omet les parenthèses pour augmenter la lisibilité):
1) Les variables-prémisse de la règle: E1= {produit, v, puht, cat, produi, taux}
2) Les variables-conséquent de la règle: E2 = {message, mois1, mois2, période}
3) Les variables-prémisse du POUR_TOUT: E3 = {nummois1, nummois2}
4) Les variables-conséquent du POUR_TOUT: E4 = {NBMOISPER, V}

On choisit au hasard une variable-prémisse de la règle, par exemple puht.
On crée la composante COMP1 du graphe. On a donc puht ∈ COMP1,
d'où produit ∈ COMP1 (lien etiq-vat avec puht)
d'où v ∈ COMP1 (lien etiq-vat avec produit),
(* fin des liens sur E1)

puis message ∈ COMP1 (lien calcul avec produit),
NBMOISPER ∈ COMP1 (lien calcul avec produit dans le conséq. définissant V),
(caht, cattc) ∈ $COMP1^2$ (lien etiq-objet avec produit),
d'où période ∈ COMP1 (lien etiq-vat avec cattc)
d'où (mois1,mois2) ∈ $COMP1^2$ (lien etiq-objet et vat-objet avec période),
d'où (nummois1,nummois2) ∈ $COMP1^2$ (lien etiq-vat avec mois1 et mois2),

On a donc la composante connexe:



Diagnostic: Le diagnostic fourni est le suivant:

```
-> Warning: La variable-premisse        produi
de la regle: ventilation_des_ventes_annuelles_d_un_produit
n'apparait pas dans le graphe des variables de la règle (Liens Simples)
-> Warning: La variable-premisse         cat
de la regle: ventilation_des_ventes_annuelles_d_un_produit
n'apparait pas dans le graphe des variables de la règle (Liens Simples)
-> Warning: La variable-premisse         taux
de la regle: ventilation_des_ventes_annuelles_d_un_produit
n'apparait pas dans le graphe des variables de la règle (Liens Simples)
-> Warning: La variable-conséquent        V
```

Il déduira:
(Ordinateur_Portable Chiffre_D_Affaire_Hors_Taxe 4500) Période (Janvier à Mars)
(Ordinateur_Portable Chiffre_D_Affaire_TTC 5292) Période (Janvier à Mars)
en supposant bien sur que l'ordinateur connait les triplets (1 Numéro_Du_Mois_De Janvier) et (3 Numéro_Du_Mois_De Mars)

```
du conséquent POUR_TOUT: ALORS1$6
de la regle: ventilation_des_ventes_annuelles_d_un_produit
n'apparait pas dans le graphe des variables de la règle (Liens Simples)
```

Le système a donc trouvé les deux erreurs présentes dans la règle. Il fournit aussi, au même niveau, les conséquences de ces erreurs, ce qui diminue la lisibilité.

. Cas particuliers reconnus et signalés comme tels:
Certaines variables ne jouent pas un rôle de variable et n'ont pas à rentrer dans ce cadre:

* les variables des conséquents CREER qui ne sont pas des variables mais le résultat de l'appel à un générateur de constantes

* les variables d'arrêt ou plus généralement de contrôle du déclenchement de règle (flag). Par exemple, la variable (statcompt) dans une règle possédant
les prémisses COMPTEUR '= (statcompt) et (statcompt) < 5
et le conséquent COMPTEUR '= (statcompt) + 1
est une variable de contrôle de déclenchement qui n'a pas forcément à être reliée aux autres variables de la règle.

C'est à ce niveau qu'ont été introduites des nuances concernant les règles-TANT_QUE et les règles-QUAND, ces règles comportant systématiquement des variables de contrôle qui nécessitent un traitement particulier.

2ème Partie: Identification des variables restantes avec des schémas connus

Nous allons ici affiner le diagnostic, en augmentant le nombre de relations valides entre variables, de manière à distinguer les cas "douteux" des cas non interprétables. L'heuristique de base de la vérification est: **"Les variables non encore reliées doivent pouvoir l'être par l'un des clichés suivants: produit cartésien, itération".**

* On parle de produit cartésien quand le seul lien existant entre deux variables (X) et (Y) définies en partie prémisse est un lien **en conséquent** de type F(X) = (Y) ou F(Z) = opérateur(X,Y, ...). Ce conséquent constitue en effet des couples ou des n-uplets contenant des objets (les instances de (X) et de (Y)) qui n'ont aucun rapport explicite entre eux, comme pour un produit cartésien. Par exemple, les deux variables (X) et (Y) sont liés par un produit cartésien dans la règle:

```
REGLE  cas_canonique_du_lien_"produit cartésien"
SI     (X) Appartient_A (E)
       (Y) Appartient_A (F)
ALORS  (X) Forme_Un_Couple_Avec (Y)
```

* On parle d'itération quand le lien entre deux variables est un lien d'écrasement de l'une (ou de son statut) par l'autre (ou par son statut).

Exemple:
```
REGLE changement_de_solution_potentielle
SI          SOLUTION_COURANTE '= (C)
            PILE_SOLUTIONS_POSSIBLES '= ((tete) LISTE (suite))
            PILE_ECHEC '= (pile2)
ALORS       SOLUTION_COURANTE '= (tete)
            PILE_SOLUTIONS_POSSIBLES '= (suite)
            PILE_ECHEC '= ((C) LISTE (pile2))
```

le lien entre C et tete d'une part, pile2 et C d'autre part est de type itération.

. Le système signale (warning/information) les variables intégrées au graphe des variables grâce à ces clichés, et fournit la liste des variables restantes.

On trouvera au 5) des exemples d'erreurs détectées grâce à ces deux modules.

c) Vérification lexicale sur les prédicats

c.1) Présentation

La partie précédente axait sa recherche d'erreurs lexicales sur les atomes en place étiq (couramment nommé objet) et vat (couramment nommé valeur). Celle-ci offre le même type de vérification mais cette fois sur les atomes en place qualité (couramment nommé attribut) et statut.

L'utilisateur est néanmoins beaucoup plus sollicité ici. En effet, le rôle des modules se limite à:

- établir une liste ordonnée des prédicats, en ventilant en fonction du type des propositions dans lesquelles apparait le prédicat considéré et des statuts qui lui sont affectés.

- souligner quelques points suspects, qui se divisent en deux catégories:
  * les singularités : utilisation unique d'un prédicat dans la base, en prémisse ou en conséquent.
  * les bizarreries :
    . mention uniquement par absence
    . destruction sans construction d'un prédicat ou sans raison apparente.

Le jugement sur la validité lexicale d'un prédicat est donc basé sur les heuristiques suivantes:

- Un prédicat doit être utilisé plusieurs fois dans la BdR: autrement dit, toute singularité orthographique est suspecte.
- Il vaut mieux se référer à ce qui existe plutôt que de miser sur ce qui n'existe pas.
- Ne détruisons que ce que nous avons construit
- Si l'on se donne la peine de détruire, c'est pour reconstruire ensuite autre chose grâce à cette destruction.

Comme précédemment, ces heuristiques n'ont pas de fondement logique. Elles ne font qu'illustrer notre vision personnelle du bon sens dans l'art de la programmation.
Nous allons maintenant détailler ces modules.

c.2) Le module ECRIPRED

1ère Partie: Création du dictionnaire des prédicats, et classification

La première partie du module crée un dictionnaire des prédicats avec leurs statuts apparaissant dans la BdR. En effet, fournir au concepteur une liste complète et structurée des objets qu'il construit est une assistance précieuse. En effet, sans aller jusqu'à parler d'improvisation complète, le concepteur découvre bien souvent en cours de construction le problème qu'il traite. Il est donc amené à créer un modèle de représentation profondément différent de celui qu'il avait en tête au départ. De là résultent des problèmes d'homogénéité et de maintien de la cohérence dans la terminologie employée. Ceux-ci seront pour la plupart résolus par un examen attentif du dictionnaire fourni ici.
On peut décrire ce traitement de la façon suivante:

. **Buts**: * Recensement des prédicats apparaissant dans la BdR
* Ventilation en fonction:
1°) du type de proposition: prémisse (PREM), conséquent normal (CONS1), conséquent écrasant (CONS2, voir la définition de <Prop-type1'> page 129)
2°) du Statut de la proposition
-> classification des prédicats d'entrée, de sortie, intermédiaires

. **Spécification du traitement**:
1°) Calcul des fonctions:
a) **Occurrence**: {PREM, CONS1, CONS2}* Pred *(ConstStatutU{(VAR)}) --> ENTIER
avec ConstStatut: l'ensemble de toutes les constantes apparaissant en statut
Pred: l'ensemble de tous les prédicats de la BdR
définie par:
Occurrence(t, p, (VAR)) = v si le prédicat p apparait v fois dans une proposition de type t et de statut variable (le terme en place statut est une variable)
Occurrence(t, p, s) = v, (avec s un objet), si le prédicat p apparait v fois dans une proposition de type t et de statut s

b) **Occurrence_Géné**: {PREM, CONS1, CONS2}* Pred --> ENTIER
définie par:

$$\text{Occurrence\_Géné}(t, p) = \sum_{s \in \text{ConstanteStatut} \cup \{VAR\}} \text{Occurrence}(t, p, s)$$

2°) Création des 3 ensembles:
Pred_Initiaux = {pred app Pred tq
Occurrence_Géné(CONS1, pred) + Occurrence_Géné(CONS2, pred) = 0}
Pred_Finaux = {pred app Pred tq Occurrence_Géné(PREM, pred) = 0}
Pred_Intermédiaires = {pred app Pred - Pred_Initiaux U Pred_Finaux }

3°) Ecriture dans le fichier de sortie:
. en considérant successivement les 3 ensembles Pred_Initiaux, Pred_Finaux et Pred_Intermédiaires
. par ordre alphabétique sur les prédicats à l'intérieur de chacun de ses trois ensembles,

. pour chaque prédicat, des nombre d'occurrences par type de proposition et par statut.

Exemple: (tiré de l'étude sur SACSO, déjà citée)

```
Nombre d'apparitions du predicat : EST_DEFINI_PAR
1) En partie premisse d'une regle ou d'un pour_tout
- avec le statut : VARIABLE, nombre d'occurrences : 20
- avec le statut : INEXISTANT, nombre d'occurrences : 6
- avec le statut : VRAI, nombre d'occurrences : 12
2) En partie consequent d'une regle ou d'un pour_tout
   a) Consequent Normaux
- avec le statut : POTENTIEL, nombre d'occurrences : 6
- avec le statut : VARIABLE, nombre d'occurrences : 4
- avec le statut : VRAI, nombre d'occurrences : 6
   b) Consequent avec Ecrasement (<--)
** Pas d'occurrence du predicat dans les consequents ecrasants **
```

2ème Partie: Détection des singularités et des bizarreries

La deuxième partie souligne les cas signalés dans la présentation de ce module, à savoir:

. les singularités orthographiques: ie les prédicats pred tels que

Occurrence_Géné(PREM, pred) = 1

ou Occurrence_Géné(CONS1, pred) + Occurrence_Géné(CONS2, pred) = 1

l'expérience montrant qu'un prédicat utilisé une seule fois (si la base de règles est suffisamment grosse) est soit inutile, soit le résultat d'un faute de frappe.

. les prédicats seulement utilisés "par absence" ie seulement présents en prémisse dans une proposition de statut INEXISTANT.

Rappelons qu'une prémisse de la forme (X) F (Y) 'INEXISTANT s'interprète:

- si (X) et (Y) sont déjà instanciés (grâce à d'autres prémisses) :

"si le triplet (X) F (Y) n'existe pas dans la base de faits courante".

- sinon:

"instancions les variables encore libres par des valeurs telles que le triplet (X) F (Y) n'existe pas dans la base de faits courante".

Donc la mention d'un prédicat uniquement sous cette forme indique une volonté de conditionner certaines actions uniquement à l'absence d'instance pour cette relation, sans jamais par ailleurs contrôler la présence ou créer certaines instances de cette relation.

Il est donc fort probable que ce prédicat ne soit que le résultat d'une déformation d'un autre prédicat qui, lui, est présent en conséquent dans d'autres règles (le fait d'utiliser en prémisse, avec le statut INEXISTANT, un prédicat jusque là déduit, est la façon - très courante - de traduire en règles le SINON du schéma SI ... ALORS ... SINON...).

Même si cette erreur s'est reproduite dans plusieurs règles à cause d'opérations de duplication, on peut de cette manière repérer (soupçonner) l'erreur.

Ces prédicats se caractérisent par:
Occurrence_Géné(PREM, pred) = Occurrence(PREM, pred, INEXISTANT)
et Occurrence_Géné(CONS1, pred) = Occurrence_Géné(CONS2, pred) = 0

. Pour des raisons similaires, on signale aussi les prédicats répondant à la signature:
Occurrence_Géné(CONS1, pred) = Occurrence(CONS1, pred, INEXISTANT)
et Occurrence_Géné(PREM, pred) = 0
En effet, on détruit en général tout ou partie des instances d'une relation afin d'empêcher certaines instanciations globales de certaines règles. Donc l'absence d'un prédicat dont on détruit des instances de toutes les parties prémisses des règles n'est pas logique et signifie très certainement une erreur.

c.3) Remarque

Ce genre de vérification en suggère très vite d'autres. En effet, même si les critères retenus pour signaler des anomalies sont pertinents, on voit bien qu'ils ne constituent pas une protection très efficace contre les erreurs de frappe et autres malfaçons qu'on a pourtant soulignées si fréquentes.

En fait, on cherche à faire un filtre "orthographique", ie capable de discerner le mot mal formé de son modèle. Formellement, cette volonté peut se traduire en terme de construction d'un module vérifiant le respect d'une bonne distance lexicographique entre les termes, ie signalant deux mots trop proches lexicographiquement pour que l'un des deux (celui le moins usité en règle générale) ne soit pas qu'une déformation de l'autre.

Exiger que les prédicats soient bien "séparés" lexicalement parait raisonnable, et prometteur. En pratique, le problème devient bien entendu beaucoup plus compliqué. Que faut-il privilégier ? Le nombre de lettres, le fait d'avoir un début commun, le nombre de lettres identiques ? On se heurte immédiatement à des problèmes proches de ceux rencontrés en reconnaissance de forme (la recherche de bons invariants) et que l'on sait particulièrement difficiles. Notre entreprise n'a donc pas voulu aller plus loin. Toutefois, certains esprits courageux et rusés auront peut-être la curiosité de mener des investigations dans cette voie ...

5) Résultats

Les vérifications décrites dans cette partie devraient être effectuées, pour être utiles, au début de la mise au point d'une base de règles. En effet, on rencontre assez vite les conséquences des erreurs détectées ici lors des premières essais de la base de règles, ce qui déclenche un effort de recherche et de réparation de l'erreur. Ce genre d'erreurs est donc rarement encore présent dans des BdR un peu "rodées", comme celles que nous avions à notre disposition. Nous n'avons donc pas détecté d'erreurs en contrôlant les BdR de l'EDF par ce module, à une exception près que nous décrirons au paragraphe suivant. Par contre, en appliquant ce contrôle à

VAUBAN1, nous avons trouvé certaines erreurs. Nous fournirons dans les pages suivantes deux erreurs détectées par GRAPHVAR dans lui-même et dans un autre module de VAUBAN1.

ECRIPRED a signalé la règle suivante du prototype de l'application SACSO:
Règle détectée:

```
REGLE analyse_DE_L_OBJECTIF_VIS_A_VIS_SITUATION_FINALE
SI
DEVELOPPEMENT-SOOUHAITABLE INHIBE_DANS (SITUATION) ' INEXISTANT
( (E) (R1) (M1) :(OBJ) ) EST_UN_OBJECTIF_DE (ACTION)
(ACTION) REALISABLE_A_PARTIR_DE ((SITUATION) ' COURANT)
((SITUATION_FINALE) ' FINALE) EST_DEFINI_PAR ((E2) (R2) (M2):(ETAT))
(SITUATION) NATURE SITUATION
(OBJ) <> (ETAT)
(OBJ) INPLIQUE (ETAT) 'INEXISTANT
ALORS
(OBJ) IMPLIQUE (ETAT)
```

Commentaire:
En effet, le prédicat IMPLIQUE est une fois orthographié INPLIQUE dans la partie prémisse de cette règle.

Cette erreur était demeurée dans cette BdR car comme la prémisse dans laquelle elle se produit a le statut INEXISTANT, elle entraînait des déclenchements supplémentaires, toujours difficiles à détecter (on détecte très mal les déclenchements excessifs d'une règle, contrairement au non-déclenchement d'une règle).

Sur cette base de connaissance, VAUBAN1 a fourni le diagnostic
Diagnostic de VAUBAN1:

```
-> Singularité orthographique:
le prédicat     INPLIQUE n'a qu'une seule occurrence
Celle-ci a lieu dans la règle:
  analyse_DE_L_OBJECTIF_VIS_A_VIS_SITUATION_FINALE

-> Mention uniquement par absence:
le prédicat     INPLIQUE n'apparait qu'en prémisse avec le statut
INEXISTANT
```

L'erreur a été reconnue comme telle par l'expert, et corrigée.

Remarques:
1) cette règle, ainsi que d'autres dans cette application, ne suit pas l'heuristique : "Dans une même règle, une variable et une constante ne doivent pas avoir le même nom". VAUBAN1 a donc détecté ici des anomalies qui n'ont pas été reconnues comme telles par l'expert.
2) cette règle contient des prémisses d'ordre deux. VAUBAN1 ignore les variables placées en position d'attribut dans ces prémisses.

1ère erreur:

Règle signalée:

```
REGLE "diagnostic_lien_complexe"
SI
    PlusieursComposantesConnexesDans(Reg) = OUI
    AppartientGrapheDesVarDe(REG) = (V) : (COUPLE)
    EstRattacheAuGrapheParLeLien(COUPLE) = (LIEN)
    Commentaire(COUPLE) = (commentaire)
    EstValidePourGrapheDe(RegOuPTout) = (REG)
    NomDeRègle(REG) = (NOMREG)
ALORS
ECRIRE("La variable "(V)" de la règle "(NOMREG)" située en "(REG)
 "est rattache au graphe par un lien de type " (LIEN))
ECRIRE((commentaire))
ECRIRE(" ")
```

Commentaire:

- on a orthographié une fois la variable (REG) sous la forme (Reg). D'où des déclenchements excessifs de cette règle, qui sont repérés par les autres règles du module auquel elle appartient.

Diagnostic de VAUBAN1:

```
-> Warning: La variable-premisse       Reg
de la regle: diagnostic_lien_complexe
n'apparait pas dans le graphe des variables de la règle (Liens Simples)

-> Warning/ ERREUR: La variable-premisse       Reg
de la regle: diagnostic_lien_complexe
n'apparait pas dans le graphe complet des variables de la règle
(Liens Simples et Complexes)
```

On a deux messages correspondant aux deux recherches successivement menées par le module GRAPHVAR. La première ne s'intéresse qu'aux liens "simples" entre variables, la deuxième tient compte des schémas du produit cartésien et de l'itération (voit la description de GRAPHVAR au 4)b))

2ème erreur:

Règle signalée:

```
REGLE "RESOUD les conflits par instanciation unique des variables
concernees"
SI
Satisfait(C) = MEME_INSTANCIATION_UNIQUE_DES_VARIABLES_CONCERNEES
VariableUn(CONF) = (X1)
VariableDeux(CONF) = (X2)
Est(CONF) = RESOLU 'INEXISTANT
ALaMemeInstanciationUniqueQue(X1) = (X2) : (U)
EstValidePour(U) = (CONF)
ALORS
Est(CONF) = RESOLU
POUR_TOUT ((N1)(N2))
TEL_QUE (NOM(R1) = (N1)
         NOM(R2) = (N2))
DEDUIRE(
ECRIRE("Le conflit "(CONF)" entre "(N1)" et "(N2)" est resolu:")
ECRIRE("les variables concernees ont meme instanciation unique")
```

Commentaire:

La variable (CONF) est orthographié (C) dans la première prémisse de la règle.

Diagnostic de VAUBAN1:

```
-> Warning: La variable-premisse       C
de la regle: RESOUD les conflits par instanciation unique des
variables concernees
n'apparait pas dans le graphe des variables de la règle (Liens Simples)

-> Warning/ ERREUR: La variable-premisse       C
de la regle:  RESOUD les conflits par instanciation unique de:
variables concernees
n'apparait pas dans le graphe complet des variables de la règle
(Liens Simples et Complexes)
```

CHAPITRE VI: Métalangage et Cohérence élargie

1 - Introduction

Nous avons annoncé au chapitre précédent que celui-ci commencerait par l'entrée du "choeur" de notre système. Nous savons que le choeur a pour rôle de de commenter l'action et de révéler les caractères et les humeurs des protagonistes. C'est une entité à la fois extérieure à l'action (elle n'intervient pas directement dans le déroulement de l'intrigue) et partie constituante fondamentale de la tragédie, médium entre le sublime et le prosaïque.

Toute proportions gardées, l'ensemble des connaissances exprimées grâce au langage que nous allons présenter dans la première partie de ce chapitre est un peu le "choeur" de la Vérification. C'est lui qui révèle les "secrets" des prédicats, orientant ainsi les traitements. C'est lui aussi qu'on enrichira dès que la phase de vérification sera terminée, un peu comme le héros va, au détour de l'action, confier au choeur ses dilemmes terribles. C'est lui enfin qui revient régulièrement relancer l'attention, où que l'on vient consulter quand on ne sait plus quoi faire.

La première partie de ce chapitre (numéroté 2)) sera donc la présentation des membres de ce "choeur". Autrement dit, nous fournirons une description complète du langage que nous avons conçu pour expliciter des connaissances que le concepteur possède sur les prédicats qu'il utilise dans sa base de règles. **On présentera ce langage comme une entité autonome, c'est-à-dire s'en s'intéresser ici à l'utilisation qui en sera faite en terme de vérification.**

La partie 3) est dédiée à la construction de la nouvelle formulation du concept de cohérence. On y décrira plusieurs types de contradictions, tant dans les faits que dans les règles, et on les situera par rapport aux propriétés énnoncées au 2). Enfin, on définera, dans ce contexte, la cohérence d'une base de connaissances.

La partie 4) décrit les travaux de vérification des propriétés des niveaux les plus bas de notre langage d'expression de métaconnaissances. Ces vérifications étant couplées à des traitements visant à découvrir de nouvelles propriétés de ces niveaux, nous rendrons compte de ces deux aspects complémentaires.

La partie 5) est la plus importante. Elle explique comment notre système vérifie les propriétés "essentielles" des prédicats. Cette partie est la plus proche des travaux recensés au chapitre 1 dans le domaine de la vérification de la cohérence.

La partie 6), très courte, contient la présentation du prolongement des méthodes exposées dans la partie précédente, pour la vérification de propriétés entre prédicats.

Enfin, nous présenterons, dans la conclusion (partie 7)), les points les plus importants développées dans ce chapitre, et nous ferons un rapide bilan des traitements décrits.

2 - Description du MétaLangage

a) Introduction

a.1) La notion de concept

Comme nous l'avons décrit au chapitre 1, notre approche de la vérification de bases de connaissances repose essentiellement sur l'utilisation d'une connaissance sur le modèle de représentation des connaissances implémentées dans le système informatique à vérifier. Cette connaissance est fournie par l'expert. Il dispose pour cela d'un langage, que nous allons détailler dans les paragraphes suivants.

Ce langage permet de préciser certaines propriétés des prédicats utilisés dans la base de connaissances. Il met en jeu deux types d'entités: les prédicats (issus de la BdR à analyser) et **les concepts**. En première approche, un concept peut être vu comme le représentant d'un ensemble de propriétés communes à un ensemble d'objets.
On va maintenant préciser d'avantage ce que nous entendons par concept, et ce qui justifie l'emploi de ce terme.

Concrètement, notre notion de concept est l'équivalent dans notre formalisme de la notion d'entité dans le modèle entité-association des bases de données: nous voulons en effet pouvoir dire que chacun des prédicats de la BdR met en relation deux concepts. Or, dans Genesia2, un prédicat met en relation deux objets GENESIA, c'est-à-dire:
- soit une constante atomique
- soit un triplet d'objets (définition récursive, cf chapitre précédent).

Donc un concept est :
- soit un ensemble d'objets du monde réel (des machines, des actions ...), une relation entre 2 objets,
- soit: . une relation[1] entre objets du monde réel,
. une relation entre un ensemble d'objets du monde réel et une relation entre deux ensembles d'objets du monde réel,
. une relation entre relations,

Dans le deuxième cas, on représentera le concept sous la forme d'un arbre binaire dont :
- chaque feuille est un concept,
- chaque noeud non terminal est un prédicat;
- la relation liant un noeud avec ses successeurs étant: "Un noeud n1 a pour fils gauche le noeud n2 et fils droit le noeud n3 si n1 met en relation les éléments du concept représenté par le sous-arbre de racine n2 avec les éléments du concept représenté par le sous-arbre de racine n3"

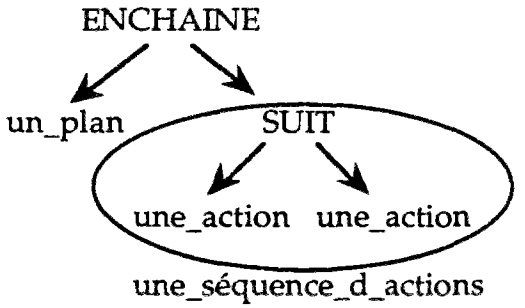
---

[1] relation est employé dans ce paragraphe dans son sens mathématique. Une relation R entre deux ensembles E et F est un ensemble de paires (x, y) avec $x \in E$ et $y \in F$, c'est-à-dire une partie du produit cartésien E * F. Une relation est donc un ensemble.

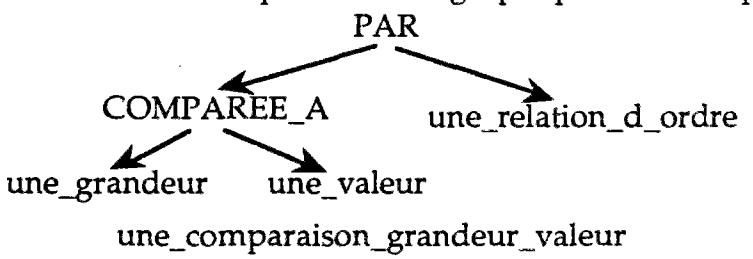
Par exemple, dans la base de connaissances de conduite de centrale (déjà citée), on manipule la notion de "séquence_d_actions" , qui désigne un couple d'actions élémentaires devant se succéder l'une à l'autre: l'action"Lancer une turbine à 1500 tours" suit l'action "Lancer une turbine à 500 tours", cette séquence intervenant dans le processus total de lancement d'une turbine. Plus précisément, dans le processus de conduite, on a identifié un certain nombre de plans, chaque plan étant constitué d'un enchaînement de séquences d'action. On a donc trois concepts:

- le concept d'action (l'ensemble des actions élémentaires),
- le concept de séquence d'actions: les séquences d'actions sont deux actions reliées par le prédicat SUIT, par exemple "Lancer une turbine à 1500 tours" suit l'action "Lancer une turbine à 500 tours". Le concept représente donc l'ensemble des instances de la relation SUIT.
- le concept de plan, un plan étant en relation, par le prédicat ENCHAINE, avec un ensemble de séquences d'action.

On a donc l'arbre suivant:



Ou encore, dans la même base de connaissances, on manipule la notion de "comparaison_grandeur_valeur", qui désigne une expression comparant une grandeur (température, pression ...) à une valeur (valeur de référence, valeur minimale admissible ...) par une relation d'ordre (SUP, INF), pour indiquer que une action EST_LA_ CONSEQUENCE_DE une_comparaison_grandeur_valeur.
On a donc une représentation graphique du concept sous la forme:



On voit donc ce qui nous distingue des notions d'entité des BdD ou de classe des Langages Orientés Objet. On pourrait sans doute parler de type abstrait, mais la notion de type renvoit à une vision statique de la description, alors que la seule caractérisation simple - mais trop générale pour notre application de la notion de concept est:
"Un concept est une application qui, à toute base de faits initiale et à tout instant de l'inférence associe un ensemble". (cf le concept de classe dans [DORMOY90] et [DORMOY91]).

a.2) L'Espace et le Temps

Nous allons décrire comment fournir à la machine un certain nombre de connaissances concernant les prédicats utilisés dans la BdR. Ces connaissances décrivent des propriétés de ces prédicats. Dès lors, se pose le problème du domaine de validité de ces propriétés. Concrètement, deux dimensions apparaissent:

- une dimension spatiale: l'espace des bases de faits initiales possibles
- une dimension temporelle: l'ensemble des moments de l'inférence. Plus précisément, on peut considérer le temps de l'inférence comme un ensemble d'entiers naturels, t = 0 étant le temps avant l'inférence, et t = n l'instant du $n^{ième}$ déclenchement de règles.

On devrait donc, en toute généralité, paramétrer les connaissances sur la BdR par deux quantités, l'une indiquant l'ensemble des bases de faits initiales concernées, l'autre l'intervalle de temps de validité de cette contrainte.
Nous n'avons pas pris en considération cet aspect spatio-temporel du problème. D'une part parce que certaines des connaissances qui serons fournies ne dépendent clairement ni du temps ni de l'espace, d'autre part, pour les autres, pour ne pas trop compliquer d'entrée le problème.
Les connaissances dont nous allons parler devront donc être valables pour toute base de faits initiale valide (ie décrétée comme telle par le concepteur) et pour tous les temps de l'inférence. La formulation de ces connaissances ne mentionnera ni le temps, ni l'espace.

Cependant, certaines de ces connaissances ne peuvent pas, au regard de notre définition du temps, être vraies "pour tout t". En effet, elles se réfèrent implicitement à un fonctionnement du moteur, qui lui permettrait de déclencher au même instant plusieurs règles ou la même règle avec des instanciations globales différentes. Puisque ce n'est pas le cas en pratique, il y aura nécessairement des intervalles de temps pendant lesquels elles seront contredites.

On aura donc deux sortes de connaissances:

- les connaissances "intemporelles", valables pour toute BFi valide et quel que soit t,
- les connaissances "quasi-intemporelles", valables pour toute BFi valide et:
  . à l'instant t = 0
  . telles que si elles sont contredites à l'instant t, elles doivent être vraies de nouveau à un instant t' voisin de t. On précisera plus bas (lors de la définition de ses propriétés, ou dans les modules de vérification) quelle interprétation précise on choisit pour la relation "voisin de".

a.3) Champs sémantiques d'un prédicat, Métabase

Chaque phrase du langage de description des métaconnaissances établit une (méta-)relation entre prédicats et concepts. Plus précisément, on aura des (méta-) relations:
. soit entre un prédicat et un concept manipulé par ce prédicat,
. soit entre prédicats utilisés dans la BdR,
. soit entre concepts.

Nous allons maintenant détailler ces métarelations. Nous insisterons particulièrement sur celles permettant de donner des informations sur la nature d'un prédicat. Ces dernières sont fournies sous la forme de triplets du type:
(un prédicat une métarelation un concept ou un mot clé du langage)
dans lequel "prédicat" désigne la relation concernée par la propriété "métarelation", à laquelle correspond l'attribut "un concept ou un mot clé du langage".
Les métarelations de cette catégorie sont appelées des **champs sémantiques**.
Et on dira que la valeur de la relation "prédicat" pour le champ sémantique "métarelation" est "un concept ou un mot clé du langage"

Rappel: les **métarelations** sont fournies sous forme de triplets GENESIA2. On parlera de la **métabase** de faits pour désigner l'ensemble de toutes les métarelations attachées aux prédicats présents dans la base de règles.

b) Connaissances sur les prédicats

b.1) Propriétés intrinsèques

Nous allons ici décrire les propriétés propres à un prédicat que notre langage permet de communiquer[2] à la machine. Elles se répartissent sur 4 niveaux, du plus élémentaire au plus abstrait. Notons que ces propriétés s'attachent toujours à un prédicat, sans mention quelconque de statut. Plus précisément, on ne distinguera jamais les propriétés du prédicat PRED quand il apparait dans un triplet avec le statut S1 des propriétés du prédicat PRED quand il apparait dans un triplet avec le statut S2[3]. On donne les propriétés de PRED valables pour toutes ses occurrences, quelque soit le statut associé aux triplets mentionnant PRED.

b.1.1) Le Niveau 1: Niveau SYNTAXIQUE

Il s'agit ici de fournir les données permettant de savoir si un triplet mentionnant le prédicat en question est "bien formé", ie si le triplet:
- met en rapport des objets de types compatibles,
- doit être imbriqué ou non dans une autre relation.

On précise donc, pour les prédicats:
a) **les concepts qu'ils mettent en relation.**
Ces concepts seront les valeurs des champs sémantiques "**Domaine Conceptuel Objet**" et "**Domaine Conceptuel Valeur**" du prédicat.

---

[2] Il faut bien sur s'entendre sur le mot "communiquer". En effet, comme nous le verrons ultérieurement, notre système n'est pas capable de (n'a pas encore les connaissances pour) utiliser toutes les connaissances fournies ici. Je crois par contre, et j'essayerai de le montrer, qu'il n'y aurait rien d'impossible à ce qu'il puisse le faire un jour, légitimant ainsi un terme qui, dans l'état actuel du système, est un peu abusif.

[3] Ce qui ne nous empêchera pas, bien entendu, de distinguer dans les règles les deux propositions.

On écrira dans la MétaBase de Faits (MBF), pour un prédicat P:

P **DomaineConceptuelObjet** cop P **DomaineConceptuelValeur** cov avec cop et cov deux concepts

Exemple:

1) Le prédicat CONTIENT est utilisé, dans la base de connaissances qui nous sert d'exemple type, pour indiquer quels équipements sont contenus dans les circuits d'une centrale électrique, établissant par exemple la relation:
Circuit_fioul CONTIENT Pompe_de_gavage

On indique alors dans la base de métaconnaissances les faits suivants:
CONTIENT DomaineConceptuelObjet un_circuit
CONTIENT DomaineConceptuelValeur un_équipement

2) Nous verrons au c) comment définir le concept "une_séquence_d_actions" dont nous parlions précédemment. Notons ici que l'on écrira pour le prédicat ENCHAINE mentionné alors, que:
ENCHAINE DomaineConceptuelValeur une_séquence_d_actions

b) la "Suffisance" du prédicat

Un prédicat est de suffisance AUTONOME s'il a un sens en elle-même, et peut donc se rencontrer sans que le triplet qui le mentionne ne soit imbriqué dans un autre triplet.
Un prédicat est de suffisance CONTEXTUELLE si la relation établie par le prédicat doit toujours être englobée dans une autre relation (on indiquera le ou les types de triplets récepteurs possibles grâce aux "relations entre concepts" décrites au c.2), car elle ne code qu'une partie d'une unité de sens donnée.
L'existence de ce genre de prédicat est due à la structure de notre langage ne permettant pas d'écrire directement des relations n-aires (avec n > 2).

On indiquera donc dans la MBF, pour un prédicat R:

R Suffisance s avec s ∈ {AUTONOME, CONTEXTUELLE}

Exemples:

1) le prédicat SUIT, déjà cité, relie deux actions ne pouvant se rencontrer sans que soit précisé l'objectif, ou les moyens, lié à ces actions successives. Il est clairement de suffisance contextuelle.

2) le prédicat "," utilisé dans les expressions de la forme:
COURS ((MATIERE) "," ((SALLE) "," ((PROF) "," ((JOUR) "," ((HORAIRE)))))) = OUI
est (caricaturalement) de suffisance contextuelle

Remarques:

1) On fournit grâce à ces deux champs sémantiques une sorte de grammaire B.N.F. des termes employés dans la base de connaissances. Une fois connu quels sont les autres relations possibles pour accompagner un prédicat de suffisance contextuelle (prédicat complémentaires, voir c.2), nous aurons les

moyens de contrôler le caractère bien-formé des faits et surtout des règles (voir 4)c)).

2) Par ailleurs, comme nous le verrons plus bas, il sera possible de définir un concept en extension. On entend par là le fait de donner intégralement l'ensemble des valeurs que le concept représente. Par conséquent, ces deux champs sémantiques peuvent permettre d'indiquer l'ensemble des valeurs autorisées en place étiq ou vat d'un triplet mentionnant le prédicat concerné, comme dans SACCO (voir [AYEL87] et [AYEL&R90]).

3) Précisons que si le sens d'un prédicat de suffisance autonome ne nécessite pas de prédicats complémentaires, cela ne veut absolument pas dire que les relations qu'il établit ne peuvent pas se retrouver dans d'autres triplets. Par exemple, on pourrait très bien écrire le triplet mentionnant qu'une proposition de type zéro plus est fausse dans un certain contexte :
(((Z) PAR (COMP)) VERIFIEE NON) DANS (CONTEXTE)
[Le prédicat PAR, de suffisance autonome, est employé à l'intérieur d'autres triplets]

4) En général, les prédicats de suffisance contextuelle résultent du choix de codage d'une relation n-aire sans utiliser une méthode de pivot, c'est-à-dire sans décomposer en n + 1 relations binaires ayant toutes le même domaine conceptuel objet. Cette partie commune sert uniquement de jointure entre les relations.

5) Les relations autonomes sont une reformulation, dans notre contexte, de la notion de classe de faits prédicatives, évoquée dans [DORMOY90].

b.1.2) Le Niveau 2: Niveau PHYSIQUE

Si l'on monte un peu sur l'échelle de la validité, il apparait qu'une fois données les propriétés relatives à chaque instance d'une relation pour qu'elle soit bien formée, il convient de préciser quels sont les rapports du prédicat, globalement, avec les concepts qu'il relie. Autrement dit, quelle est la nature physique du rapport du prédicat avec les concepts qu'il manipule.

Plus précisément, on va indiquer, pour un prédicat donné, les deux informations suivantes:

a) Recouvrement

On dira qu'un prédicat **recouvre totalement** son concept objet (resp. valeur), si pour chaque instance de ce concept, il existe une instance de la relation utilisant ce fait. Sinon on dira que le prédicat **recouvre partiellement** le concept concerné.

On écrira ces informations dans la MBF de la façon suivante, pour un prédicat R:

R RecouvrementObjet ro
R RecouvrementValeur rv
avec **(ro, rv)** $\in$ **{TOTAL, PARTIEL}^{2}**

Cette propriété est **quasi-intemporelle**. Elle doit être respectée par toute base de faits initiale, et toute création d'un élément du concept "totalement recouvert" par

R doit être suivie par sa mise en relation par R avec un élément adéquat (voir l'exemple et la note ci-dessous).

Exemple:

Tous les circuits d'une centrale contiennent des équipements. Les équipements d'un circuit lui sont assignés par l'intermédiaire du prédicat CONTIENT. On écrit par exemple: circuit_fioul CONTIENT pompe_de_gavage

Donc le recouvrement du concept un_circuit par CONTIENT est total. Ce qu'on écrit : CONTIENT RecouvrementObjet TOTAL[4]

Par contre, certains équipements apparaissent ailleurs que dans des circuits, et ne pas sont mis en relation avec l'entité les contenant par l'intermédiaire de CONTIENT. Donc le recouvrement de un_équipement par CONTIENT est partiel. Ce qu'on écrit : CONTIENT RecouvrementValeur PARTIEL

b) Type Physique

Après avoir indiqué le rapport "global" de la relation avec ses concepts, on va maintenant s'intéresser à la nature des liens existants entre les instances de cette relation. Ces informations sont précisées dans le champ sémantique "**Type Physique**"du prédicat. sous la forme (pour un prédicat R):

R TypePhysique tp avec tp ∈ {FONCTIONNEL, INJECTIF, ANTISYMETRIQUE, SYMETRIQUE, REFLEXIF A DROITE, REFLEXIF A GAUCHE}

Précisons le sens de ces propriétés. On dira que R possède le type physique :

* **fonctionnel** , ou encore **fonctionnel à droite**, si[5]:
  $x\,R\,y1$ et $x\,R\,y2 \Longrightarrow y1 = y2$

* **injectif** , ou encore **fonctionnel à gauche**, si:
  $x1\,R\,y$ et $x2\,R\,y \Longrightarrow x1 = x2$

* **antisymétrique** si:
  $x\,R\,y$ et $y\,R\,z \Longrightarrow z <> x$ [6]

---

[4] Donc, a) si "CONTIENT" apparait en BFi, tous les éléments de "un_circuit" doivent être reliés par CONTIENT à des éléments de "un_équipement"

b) si les instances de "CONTIENT" sont déduites par les règles,

b.1) si on ne crée pas dans l'inférence des éléments de "un_circuit", les instances de "CONTIENT"doivent toutes être créées "en même temps", c'est-à-dire par déclenchements successifs d'une ou plusieurs règles (voir le fonctionnement du moteur d'inférences) de telle sorte qu'à la fin de l'exploitation de cette règle ou de ce paquet de règles, tout élément de "un_circuit" soit relié par CONTIENT à un élément de "un_équipement"

b.2) si on crée dans l'inférence des éléments de "un_circuit", cette création soit immédiatement suivie de la mise en relation par CONTIENT avec un équipement grâce au déclenchement d'une des règles créant les instances de"CONTIENT", ce mécanisme rappelant celui de l'exploitation des règles de completion de SACCO([AYEL&al88]), ou des règles du réseau adjoint de SUPER ([FONT&al88]).

[5] Dans les formules qui vont suivre , x, y1, y2, z sont des variables "mathématiques" .

[6] Notre définition diffère de celle des mathématiques, qui est: (AS): $x\,R\,y$ et $y\,R\,x \Rightarrow x = y$

* **symétrique** si:
x R y ==> y R x [7]

* **réflexif à droite** (resp. **réflexif à gauche**) si:
x R y ==> y R y (resp. x R x) [8,9]

Remarque sur les prédicats de suffisance contextuelle:

Dans le cas où le prédicat est de suffisance CONTEXTUELLE, ces propriétés doivent être réinterprétées en tenant compte du fait que le prédicat apparait systématiquement en tant que objet ou valeur d'une ou plusieurs autres relations (appelées prédicats complémentaires , voir c.2).

Par exemple, si R est de type physique fonctionnel et de suffisance contextuelle, et si G est un prédicat complémentaire de R, suivant la place de la relation R dans la relation G, la propriété s'écrit:

(x R y1) G z et (x R y2) G z ⇒ y1 = y2

Ou z G (x R y1) et z G (x R y2) ⇒ y1 = y2

Comprenons que (x F y1) G z1 et (x F y2) G z2, avec y1 <> y2 sont tous les deux possibles si z1 <> z2.

Plus généralement, si le prédicat est de Suffisance CONTEXTUELLE et possède le type physique tp, la propriété que possède R s'obtient à partir de la formule définissant tp mais en remplaçant tous les termes de la forme u R v par:

- (u R v) G z si G est un prédicat complémentaire de R utilisant cette relation en place objet
- z G (u R v) si G est un prédicat complémentaire de R utilisant cette relation en place valeur (vat)

Exemples de relations possédant des types physiques particuliers:

- Dans notre modèle de centrale,

1) le prédicat CONTIENT, reliant les circuits aux équipements que ces circuits contiennent, est de type physique injectif. Chaque équipement appartient à un circuit donné, celui-ci en contenant plusieurs.

2) le prédicat SUIT possède le type physique "FONCTIONNEL". Par ailleurs, on sait qu'il est de suffisance contextuelle (il sert à indiquer la succession de deux actions dans un plan donné). On est donc dans le cas d'application de la

---

et qui permet donc d'avoir des triplets de la forme a R a, ce qui n'est pas possible avec notre définition. La notion d'antisymétrie que nous avons définie est donc une antisymétrie "stricte", c'est-à-dire réunissant l'antisymétrie et l'irréflexivité mathématiques.

[7] voici un autre cas de quasi-intemporalité. Le moteur ne peut pas déduire les deux instances en même temps. Mais puisque le prédicat est déclaré symétrique par le concepteur, la déduction de l'instance symétrique doit pouvoir être faite juste après, c'est-à-dire lors dans la même phase de d'exploitation de la règle (voir dans la partie vérification).

[8] Idem ci-dessus

[9] Là encore, nos définitions diffèrent de la définition des mathématiques ((RE): $\forall x \in E, x R x$), et se rapprochent beaucoup plus de la double propriété mathématique : Symétrie + Transitivité. En effet dans ce cas, on a: x R y ⇒ y R x ⇒ x R y et y R x ⇒ x R x (réflexivité à gauche)
et x R y ⇒ y R x ⇒ y R x et x R y ⇒ y R y (réflexivité à droite)

remarque ci-dessus. Ici, la formule après modification signifie qu'une action SUIT de manière unique une autre action pour un plan donné, mais peut très bien être suivie de tout autre chose si le but poursuivi est différent.
3) le prédicat SUIT possède également le type physique "ANTISYMETRIQUE".

- Dans un cadre quelconque:
Supposons que des ensembles soient construits de telle sorte qu'ils aient leurs intersections deux à deux toutes identiques. Cette intersection, notée F est donc l'intersection globale de tous ces ensembles: $F = \bigcap_{i \in (1,\ldots,n)} E_i = E_i \cap E_j, \forall i, j$

Donc $\forall i$, si $x \in F$, alors $x \in E_i$, et si $x \in E_i - F$, alors $\forall j \neq i$, $x \notin E_j$

Admettons alors que pour des questions de gain de place mémoire, on ait écrit:
pour indiquer que x appartient à tous les ensembles E_i : x ElementDe F, ,
pour indiquer que x appartient uniquement à $E_i - F$: x ElementDe E_i
[si on a 100 éléments appartenant à l'intersection de 50 ensembles, on créera ainsi les 100 instances indiquant l'appartenance de ces éléments à F plutôt que d'écrire les 5 000 appartenances de ces éléments à tous les E_i.]

Maintenant, pour savoir si un élément appartient à E_i, ne disposant pas de OU dans notre langage, on va créer une nouvelle relation, BonPour, et associer à chaque ensemble E_i les deux instances de BonPour:
Ei BonPour Ei et F BonPour Ei.
et remplacer dans les règles la prémisse: (X) ElementDe (Ens)
par les deux prémisses: (X) ElementDe (Ens)
(Ens) BonPour (Ei)
Le prédicat BonPour est donc réflexif à droite.
[On crée donc 100 faits supplémentaires, à comparer aux 4 900 économisés]
Cet exemple est tiré de VAUBAN1. Les ensembles Ei sont les conflits entre règles (voir plus loin). Quand dans le cadre d'un conflit, on examine la valeur d'un terme atomique, celle-ci peut être connue de tous les conflits si ce terme est une constante, ou dépendante du contexte du conflit, si ce terme est une variable de l'une des règles du conflit]

Remarques sur des valeurs particulières du champ sémantique "TypePhysique:
1) Les deux premiers valeurs du champ type physiques, ainsi que la notion de recouvrement, rejoignent les cardinalités mini et maxi associées à un lien entité-association dans un modèle conceptuel de BdD.
Plus précisément, si on a dans le modèle conceptuel et pour une relation $\mathcal{R}$ entre E et F le schéma suivant:
E ————min1,max1———— $\mathcal{R}$ ————min2,max2———— F
on a les correspondances:
$\mathcal{R}$ RecouvrementObjet TOTAL / PARTIEL $\Leftrightarrow$ min1 > 0 / min1 = 0
$\mathcal{R}$ RecouvrementValeur TOTAL / PARTIEL $\Leftrightarrow$ min2 > 0 / min2 = 0
$\mathcal{R}$ TypePhysique INJECTIF $\Leftrightarrow$ max2 =1
$\mathcal{R}$ TypePhysique FONCTIONNEL $\Leftrightarrow$ max1 =1

2) La propriété d'antisymétrie n'est évidemment intéressante que si les domaines conceptuels objet et valeur du prédicat ont une intersection non vide, sinon elle est trivialement vraie. Dans le même ordre d'idée, les propriétés de symétrie et de réflexivité nécessitent que les domaines conceptuels objet et valeur soient égaux.

3) On pourrait de même introduire la notion d'irréflexivité, de transitivité etc... N'en ayant pas eu besoin jusqu'à présent, nous n'avons pas traité ces cas.

REMARQUES GENERALES sur le champ sémantique "TypePhysique":

1) La relation TypePhysique est multivaluée. Un même prédicat peut en effet très bien être, par exemple, à la fois une fonction à gauche et une fonction à droite (c'est alors une bijection).

2) Si l'on compare ces propriétés à celles exprimables grâce à des contraintes d'incohérences, au sens COVADIS, SACCO, COCO-X [LOISEAU90], ou TIBRE ([LALO89]), on s'apercoit:

. **que nous sommes en partie moins général** dans la mesure où les contraintes qui correspondent à nos propriétés ne comportent au plus que 3 termes (et non un nombre quelconque comme dans une règle d'incohérence). En particulier, la prise en compte de contextes quelconques, impossible ici, pourrait l'être par COCO (cf [LOISEAU90]).

. **mais que nous sommes aussi plus général.**

Par exemple, la propriété de symétrie ne s'écrit pas sous la forme d'une contrainte de cohérence. En effet, une contrainte de cohérence est une expression de la forme : Pred1(x1, y1, ...) et Pred2(x2...) ... et Predk(xk, ...) $\Rightarrow \perp$ ie une conjonction d' expressions ne contenant pas de négation.

Or la symétrie s'écrit: R(x, y) et NON(R(y, x)) $\Rightarrow \perp$

La différence est en fait de taille: **si vérifier une contrainte de cohérence revient toujours à prouver l'indéclenchabilité d'une règle dans un contexte donné**, ie prouver une incompatibilité entre un contexte et les prémisses d'une règle, **vérifier la symétrie d'une relation consiste au contraire à prouver la déclenchabilité systématique d'une règle sous certaines conditions**, ie à faire une simulation symbolique du travail du moteur d'inférence, ce qui est beaucoup plus difficile.

Toutefois, ce type de propriétés était déja présent dans SACCO ([AYEL&al88]), ou SUPER ([FONT&al88]), sous la forme "opérationnelle" de règles de complétion (dans SACCO) ou de règles du réseau adjoint (dans SUPER), ces règles étant exploitées, parallélement à celles de l'expertise, pour compléter les déductions effectuées par ces dernières.

b.1.3) Le Niveau 3: Niveau MATHEMATIQUE

Ce niveau concerne, comme le niveau 2, les relations entre les différentes instances d'une relation. La différence avec le niveau précédent est qu'on décrit ici les propriétés du prédicat à un niveau purement conceptuel, c'est-à-dire sans tenir compte de l'implémentation réalisée dans la base de règles: en effet, il est très fréquent d'aboutir, pour des raisons techniques (gain d'espace mémoire, efficacité de l'algorithme d'instanciation ...), à des implémentations respectant d'autres lois mathématiques que celles que la sémantique du prédicat indiquait de manière évidente. On indiquera donc ici les propriétés "conceptuelles" des prédicats. Ces contorsions ont des conséquences importantes sur l'écriture des règles, et c'est pour cela que nous nous y intéressons.

On écrira dans la MBF, et si cela est différent de ce que l'on a affirmé au niveau 2), pour un prédicat R:

R **TypeMathématique**[10] prop
avec prop ∈ {**FONCTIONNEL, INJECTIF, ANTISYMETRIQUE, SYMETRIQUE, REFLEXIF A DROITE, REFLEXIF A GAUCHE**}
∪ {**FONCTION_UNAIRE**}

pour indiquer que, conceptuellement, le prédicat R possède la propriété prop.

Les valeurs associées à ce type sémantique sont, sauf la dernière, identiques à celles associées au champ TypePhysique. La dernière est spécifique à ce niveau. Avant de voir ce que nous attachons comme sens à cette valeur, illustrons l'intérêt de l'expression de ce type de connaissances par des exemples.

Exemples:

1) Supposons que nous réalisions un système qui manipule des faits représentant des règles (comme le nôtre). On va donc manipuler des variables - instanciées ou non, et des constantes. Très vite deviendra nécessaire de marquer les couples d'objets apparemment indifférenciés que l'on a pu, par déduction, distinguer. On va créer la relation "<>" dans ce but. Elle est clairement symétrique. Mais, pour gagner un facteur 2 sur le nombre de faits à gérer, on ne créera une instance de cette relation que si l'instance symétrique n'existe pas. Concrètement, on ne génère: V "<>" W {Dans contexte ...}
que si l'on n'a pas déjà déduit: W "<>" V {Dans contexte ...}

Bien entendu, la contrepartie de cette économie de place mémoire sera d'augmenter le nombre de règles nécessaires au traitement de cette relation. Par exemple, au lieu d'avoir une seule règle de propagation, du type:

REGLE r1: SI V "<>" W ET Instanciation(V) = a {Dans contexte ...}
ALORS Instanciation(W) <> a {Dans contexte ...}

on devra écrire également:

REGLE r1': SI V "<>" W ET Instanciation(W) = a {Dans contexte ...}
ALORS Instanciation(V) <> a {Dans contexte ...}

---

[10] Notre terminologie désignerait donc les mathématiques comme le domaine de conceptualisation de phénomènes physiques. Qu'on nous pardonne ...

La vérification de la présence de ce genre de symétrie locale dans la base de règles pourra s'effectuer dès que l'on sait que:
"<>" TypePhysique ANTISYMETRIQUE, "<>" TypeMathématique SYMETRIQUE:

2) Voici un exemple de prédicat ayant un Type Physique quelconque (ie ne relevant pas de l'un des schémas donnés au niveau 2) et un TypeMathématique FONCTIONNEL.

Dans le système de contrôle de centrale, on associe à chaque matériel, à un instant donné, un état. Ceci se fait par l'intermédiaire du prédicat EST_DANS_ETAT:
exemple: Alternateur EST_DANS_ETAT Couplé_à_vide
Ce prédicat est donc clairement fonctionnel (et potentiellement non monotone, un matériel pouvant changer d'état). Pourtant, la représentation physique de ce prédicat est une relation multivaluée. On peut trouver par exemple simultanément les faits:

Alternateur EST_DANS_ETAT Couplé_à_vide
Alternateur EST_DANS_ETAT Découple
Alternateur EST_DANS_ETAT Montée_de_charge_à_1mw/mn

En fait, le concepteur n'a pas eu besoin de distinguer ici l'état courant du matériel (qui est unique à un instant donné) des états potentiels, cette distinction s'effectuant par ailleurs (le matériel est dans l'état déterminé par l'action se situant elle-même dans la phase courante d'exploitation). L'implémentation n'est donc pas une fonction, bien que conceptuellement elle en soit une.

La dernière valeur possible, FONCTION_UNAIRE, indique que le prédicat, bien qu'étant syntaxiquement une relation binaire, est conceptuellement un symbole fonctionnel unaire.

Par exemple, supposons que l'on écrive un système qui génère des hypothèses, puis les considère dans un ordre qu'il détermine. On peut alors avoir besoin d'un prédicat permettant de distinguer les hypothèses envisagées de celles qui sont encore en attente d'examen. Appelons ce prédicat EST_EXAMINE. Ces instances seront toutes de la forme: une_hypothèse EST_EXAMINE OUI, et on testera en prémisse de règle si l'hypothèse (H) est examinée ou non en écrivant:
(H) EST_EXAMINE OUI {'INEXISTANT}
Le triplet "H1 EST_EXAMINE OUI" doit donc être interprété: l'hypothèse H1 possède la propriété "EST_EXAMINEE". Ce prédicat est donc, conceptuellement, un symbole fonctionnel unaire.

Plus généralement, les prédicats de TypeMathématique FONCTION_UNAIRE sont des prédicats mettant en relation des instances d'un concept avec une constante unique (du type OUI, VRAI, VU ...). La constante n'a pas d'importance. C'est le fait d'avoir la propriété ou pas que l'on testera. Autrement dit, les occurrences dans des règles sont de la forme (on prend par exemple le prédicat EST_RESOLU):
EST_RESOLU(X) = OUI ou EST_RESOLU(X) = OUI 'INEXISTANT
On aimerait écrire EST_RESOLU(problème_xx), mais le langage ne le permet pas.

b.1.4) Le Niveau 4: Niveau FONCTIONNEL

Ce niveau a pour vocation de recueillir une information quant au rôle du prédicat dans la base de connaissances. En effet, nous nous sommes rendus compte que, dans des BdR de vocations très différentes, revenaient souvent certaines structures de capture de la connaissance. D'où l'idée de ranger, quand c'est possible, les prédicats dans des "boites" se référant à leur fonction dans le processus déductif.

On écrit donc dans la MBF, pour un prédicat R:

R **TypeFonctionnel** prop avec
prop ∈ { **ORDONNATEUR, COORDONNATEUR** (FAIBLE, NORMAL, STRICT),
OPERATEUR_DE_DISJONCTION,
LIEN_CONTEXTUEL (SITUATIF, LOCATIF, INSTRUMENTAL, CAUSAL, BENEFACTIF),
PREDICAT_DE_SUBSTITUTION, CURSEUR, ATTRIBUT, ASSOCIATION}

Cette liste n'est pas exhaustive. Elle correspond aux rôles que nous avons distingué jusqu'à présent, et qui sont les suivants :

+ ORDONNATEUR, COORDONNATEUR:

Ce sont des prédicats ayant pour but d'ordonner, de classer, les éléments d'un ensemble (en occurrence les instances du concept objet de la relation). On distingue:

1°) Les relations d'ordonnancement, que l'on peut définir de la manière suivante:
$\mathcal{R}$: E ---> Ensemble numérique : N, Z, N^2
avec E un concept donné (par exemple, un ensemble de taches)
N l'ensemble des entiers naturels, Z celui des relatifs
Ces relations servent à établir un ordre (par exemple de traitement) sur lés éléments de l'ensemble E

Exemples de prédicats de ce type:

- les prédicats qui associent un numéro aux noeuds d'un graphe:
dans un graphe potentiel-tache par exemple
tache_convocation_expert APourPlace 7

- les prédicats NuméroRegle et NumeroConsequentCR de notre application, qui associent aux "djinns" représentant respectivement une règle et un conséquent de règle sa place dans la base de règles (respectivement dans la partie conséquent de la règle):
R15 NumeroRègle 15
(ALORS2$3 Consequent R2) NumeroConsequentCR 3

2°) Les relations de coordonnancement, que l'on définit de la manière suivante:
R est une relation de coordonnancement si R est une partie de E^2 (ou si l'on préfère une relation de E dans E), qui, par sa structure, induit un ordre sur E.
L'ordre peut être:

- faible : la relation est réflexive et transitive, ce qui correspond en mathématique à la notion de préordre,

- normal : la relation est réflexive, transitive et antisymétrique,
- strict : la relation est irréflexive et transitive

Exemples de prédicats de ce type:

- les prédicats établissant des précédences ou des enchaînements entre taches (plus généralement décrivant les arcs d'un "bon" graphe, ie un graphe sans cycle), comme le prédicat SUIT de l'application de conduite de centrale:
"Lancer une turbine à 1500 tours" SUIT "Lancer une turbine à 500 tours"
ou
Noeud_"*" FILS Noeud_"+"

+ OPERATEUR DE DISJONCTION

C'est un prédicat n'ayant pas d'autre fonction que d'implémenter un OU logique. En effet, ne pouvant pas directement écrire en Génésia2
SI F(X) = a1 OU F(X) = a2 OU ... OU F(X) = an

- soit on écrit les n règles,
- soit on introduit un prédicat supplémentaire et on écrit:
  - une seule règle sous la forme

SI F(X) = (Y) et VALIDE(Y) = OUI

- les n faits: a1 VALIDE OUI, a2 VALIDE OUI, ... , an VALIDE OUI

le prédicat rajouté (ici VALIDE) est appelé Opérateur de Disjonction.

Par exemple, le prédicat BonPour utilisé dans l'exemple illustrant la notion de réflexivité, au b.1.2) est un opérateur de disjonction.

+ LIEN_CONTEXTUEL

Ce sont des prédicats servant à compléter l'information principale contenue dans l'élément se trouvant en place objet dans la relation , en indiquant un des éléments suivants: on distingue les liens contextuels Situatifs (précisant un lieu, exemple: AU, DANS), Locatifs (précisant un temps, exemple: A) Instrumentaux (précisant un instrument, exemple: AVEC), Causal (précisant la cause, exemple : PARCE QUE), Bénéfactif (précisant le bénéficiaire, exemple: POUR)

Par exemple, dans l'expression (issu de VAUBAN1, voir la partie résultat du chapitre précédent):

```
ALaMemeInstanciationUniqueQue(X1) = (X2) : (U)
EstValidePour(U) = (CONF)
```

(avec X1 et X2 représentant des variables et CONF représentant un conflit entre règles)
le prédicat EstValidePour est un lien contextuel situatif.

+ PREDICAT_DE_SUBSTITUTION

Ce sont des prédicats qui, par exemple dans un raisonnement hypothétique, permettent les retours arrières. Plus précisément, il est fréquent qu'à un instant donné, on duplique un prédicat construit ou utilisé jusque là et que l'on se serve de la copie pour effectuer un traitement entraînant la destruction partielle de ce prédicat. Le prédicat-sosie est appelé prédicat de substitution. Il doit "hériter" de

tout ou partie des propriétés du prédicat modèle. C'est seulement pour ce genre d'opérations qu'on acceptera des règles du style:
SI F(X) = (Y) ALORS G(X) = (Y)

+ CURSEUR

Ce type désigne un prédicat dont le rôle est de distinguer tour à tour tous les éléments d'un ensemble. Plus précisément, à tout instant de l'inférence, il n'y a qu'un seul triplet le mentionnant (il n'établit qu'une seule relation), et l'inférence le fera, dans un ordre donné, parcourir tout l'ensemble des instances de son concept objet. C'est donc un prédicat de type mathématique "fonction_unaire", fondamentalement non monotone, étant "incrémenté" à la fin d'une boucle dans une base de règles et ne devant l'être nulle part ailleurs.

Exemple de prédicat de ce type:

Tous les prédicats du type "Machin"_Courant, par exemple le prédicat VARIABLE_COURANTE, désignant, parmi toutes les variables que gère le résolveur de problèmes combinatoires SIREN [LUCAS90], celle qui doit maintenant être considérée pour le traitement (recherche des bornes, écriture de la boucle, propagation sur les bornes des variable suivantes).

+ ASSOCIATION, ATTRIBUT

En linguistique, on distingue le rapport actif et le rapport attributif, selon que les deux éléments en rapport sont tous les deux "actants" ("le chat griffe l'enfant"), ou si un seul des deux est actant ("le chat est noir").
Sont déclarés respectivement attribut et association tous les prédicats ne rentrant pas dans une des catégories ci-dessus, suivant qu'ils mettent en rapport deux éléments dont l'un (l'objet) "a pour attribut" l'autre (la valeur), ou que l'on ne peut établir cette hiérarchie.

Ce niveau de description n'est pas utilisé par notre système dans son état actuel de développement. Ce qui ne veut certainement pas dire qu'il est intrinsèquement inutile. En effet, il pourrait assez rapidement permettre de vérifier la pertinence des propriétés mentionnées aux niveau inférieurs. Il existe en effet des relations fortes entre les valeurs possibles à ce niveau et les propriétés possibles aux niveaux inférieurs. Par exemple:

* si R est, fonctionnellement, un lien conceptuel, alors si on note co et cv respectivement ses domaines conceptuels objet et valeur, alors

- soit co, soit co (mais pas tous les deux) doit être un concept non élémentaire, c'est-à-dire doit représenter les instances d'une relation (heuristique).
- celui qui est un concept non-élémentaire doit être recouvert totalement par R,
- R doit établir un lien univoque entre les éléments du concept non élémentaire et leur contexte. Il doit donc avoir un type physique (ou mathématique) fonctionnel ou injectif.

* si R est, fonctionnellement, un opérateur de disjonction, et si on note co et cv respectivement ses domaines conceptuels objet et valeur, alors:

- soit co, soit cv (mais pas tous les deux) doit représenter un ensemble fini de valeurs (voire un singleton),

- l'autre concept doit lui représenter au moins l'union de deux concepts (par définition) ou un n-uplet (avec $n \geq 2$) de valeurs,

- comme dans le cas du lien contextuel, le lien doit être univoque et recouvrir totalement le concept représentant une union de plusieurs concepts.

Les connaissances fournies à ce niveau pourraient donc très vite servir à vérifier la cohérence de la description, ou à simplifier celle-ci, la valeur donnée à ce niveau de description déterminant une partie de celles à fournir aux niveaux inférieurs.

REMARQUES:

1) Cette partie a été largement inspirée par notre incursion, sans doute un peu brève, dans le domaine de la Linguistique. Nous voulions tirer profit des travaux sur la compréhension de la structure de langages très compliqués (les langues naturelles) pour notre problème de "compréhension" de langages triviaux (les langages informatiques).
Plus précisément, mis-à-part les différents cas de liens contextuels, inspirés de [SAUCET83], cette étude est issue de l'examen de [POTTIER85].

2) Si les niveaux 1 et 2 de notre langage s'apparentaient aux schémas conceptuels des BdD, ce niveau a un lien "spirituel" avec le concept de réseaux sémantiques. En effet, il est clair que nous avons en ligne de mire une théorie rigoureuse de la déduction, en s'appuyant sur une discrétisation des rapports possibles entre concepts, ie du rôle des prédicats.

b.2) Relations entre prédicats

Après avoir établi dans la partie précédente le modèle d'expression des connaissances sur la nature intrinsèque d'une relation, nous allons présenter le cadre permettant d'expliciter des liens entre deux relations. Ces liens se divisent en trois groupes, les relations de synchronisme, de complémentarité et d'exclusion.

b.2.1) Les Relations de Synchronisme

On parle de relation de synchronisme entre deux prédicats R1 et R2 quand l'existence d'une instance pour R1 entraîne l'existence d'une instance pour R2.
Ce sont donc des propriétés quasi-intemporelles (cf a.2)), que l'on interprètera ici comme au b.1.2).

Plus précisément, on dira que:

+ P et Q sont **synchrones-objet** (resp. **synchrones-valeur**) ssi:

$\forall x$, si x P y , alors $\exists z$ tq x Q z (resp. $\forall x$, si y P x, alors $\exists z$ tq z Q x)

+ P et Q sont **synchrones-inverse**

amont si:$\forall x$, si x P y , alors $\exists z$ tq z Q x,

aval si: $\forall x$, si x P y, alors $\exists z$ tq y Q z

+ P et Q sont **synchrones-triplet**

objet si:$\forall x,y$ x P y $\Rightarrow$ $\exists z$ tq (x P y) Q z,

valeur si:$\forall x,y$ x P y $\Rightarrow$ $\exists z$ tq z Q (x P y)

on écrira ces relations dans la MBF de la façon suivante, respectivement:

P **SynchroneObjet** Q , P **SynchroneValeur** Q
P **SynchroneInverseAmont** Q , P **SynchroneInverseAval** Q
P **SynchroneTripletObjet** Q , P **SynchroneTripletValeur** Q

Exemples:

1) Dans notre système, certains conflits entre règles peuvent se résoudre sous certaines conditions sur deux variables précises des règles concernées. On a donc deux prédicats, VariableUn et VariableDeux, qui relient l'identificateur du conflit aux deux variables concernées. Ces prédicats sont synchrones-objet.

2) Dans le système de conduite de centrale, on a le schéma suivant:
une action donnée agit sur un matériel, ce qui augmente, diminue, ou fixe des grandeurs. Donc on définit simultanément les deux objets GENESIA:
une_action AGIT_SUR un_materiel
(par exemple: Ouvrir AGIT_SUR Vannes_turbines), et
((une_action AGIT_SUR un_materiel) INFLUE_SUR une_grandeur) EN un_taux [11]
(par exemple: ((Ouvrir AGIT_SUR Vannes_turbines)
INFLUE_SUR Pression_circuit_vapeur) EN "+")

On a donc : AGIT_SUR et INFLUE_SUR synchrones-triplet-objet
(et INFLUE_SUR Suffisance CONTEXTUELLE).

---

[11] Le concept un_taux est défini en extension (voir c.1) comme l'ensemble {+, -, 0}. "+" signifie que l'action sur le matériel augmente la grandeur, "-" qu'elle la diminue,"0" qu'elle la fixe.

Remarques:

1) Ces relations ne sont pas symétriques: par exemple, si P est synchrone-objet avec Q, Q n'est pas forcément synchrone-objet avec P.

2) Cette propriété est trivialement acquise dès lors que les deux prédicats ont un recouvrement TOTAL de domaines conceptuels ad hoc: par exemple, P est synchrone-objet avec Q si:
DomaineConceptuelObjet(P) = DomaineConceptuelObjet(Q) et
RecouvrementObjet(P) = RecouvrementObjet(Q) = TOTAL

3) On notera dans l'exemple la différence entre synchrone triplet et dépendance sémantique (cf champ Suffisance du Niveau 2). Dans le cas de deux relations synchrones triplets, le triplet intérieur existe de façon autonome. On lui associe seulement un autre triplet, qui l'englobe.

b.2.2) Les Relations de Complémentarité

On parle de relation de complémentarité entre deux relations R1 et R2 quand la non-existence d'une instance pour R1 entraîne l' existence d'une instance pour R2. Ce sont aussi des propriétés quasi-temporelles (voir a.2) et b.1.2)).

Plus précisément, on dira que P et Q sont **complémentaires-objet** (resp. **complémentaires-valeur**) ssi:

DomaineConceptuelObjet(P) = DomaineConceptuelObjet(Q)

et $\forall x$ instance de DomaineConceptuelObjet(P), $\exists y$ tq $x\ P\ y$ ou $x\ Q\ y$

(resp. DomaineConceptuelValeur(P) = DomaineConceptuelValeur(Q)

et $\forall x$ instance de DomaineConceptuelValeur(P), $\exists y$ tq $y\ P\ x$ ou $y\ Q\ x$)

Donc, toute instance x du concept objet de P (ou de Q), possède soit une "image" par P, soit par Q. La relation P $\vee$ Q recouvre totalement "son" domaine conceptuel objet.

On écrira ces relations dans la MBF de la façon suivante, respectivement:

P **ComplémentaireObjet** Q
P **ComplémentaireValeur** Q

Exemples:

1) Dans le système expert (décrit dans [LAFON87]) qui vérifie la cohérence de BdR[12] d'ordre Zéro Plus, on utilise le concept "proposition" pour désigner une expression de la forme "Entité Comparateur Valeur". Ces "propositions" peuvent apparaître en prémisse ou en conséquent des règles de la base de règles que l'on vérifie. On a donc deux relations reliant les "propositions" aux règles dans lesquelles elle apparaissent. Par exemple, prop12 PREMISSE R15 signifie que la proposition désigné par prop12 apparait en prémisse de la règle désigné par R15, et prop23 CONSEQUENT R15 signifie que la proposition prop23 apparait en conséquent de la règle R15.La description (partielle) de ces deux prédicats est: PREMISSE DomaineConceptuelObjet une_proposition,
CONSEQUENT DomaineConceptuelObjet une_proposition
et PREMISSE Complémentaires-Objet CONSEQUENT

---

[12] abréviation de Base de Règles

2) Dans le système de conduite de centrale, un couple grandeur-valeur peut être soit une contrainte (prédicat EST_UNE_CONTRAINTE), soit un objectif (prédicat EST_UN_OBJECTIF) d'une action sur un matériel:
EST_UNE_CONTRAINTE et EST_UN_OBJECTIF sont donc complémentaires-objet.

Remarques:

1) On a défini la complémentarité de manière binaire.Il faudrait mieux la définir de façon n-aire. On pourrait alors dire par exemple qu'une action est soit la première (prédicat PREMIERE_DE), soit la dernière (prédicat DERNIERE_DE), soit une action intermédiaire (prédicat INTERMEDIAIRE_DE), d'un plan.

2) Cette propriété est trivialement vraie si on sait que les deux prédicats ont des recouvrements totaux d'un même domaine conceptuel.

3) La notion de complémentarité a son équivalent dans le domaine des bases de données sous le nom de contraintes structurelles de totalité (d'après [COMBE91], chap. 3: "Classification des contraintes d'intégrité, p 45).

b.2.3) Les Relations d'Exclusion

On parle de relation d'exclusion entre deux prédicats R1 et R2 quand l'existence d'une (forme donnée) d'instance pour R1 entraîne la non-existence d'une forme donnée d'instance pour R2.

Plus précisément, on dira que:

+ P et Q sont **différents-objet** (resp. **différents-valeur**) ssi:

$\forall x,\ x\,P\,y$ et $x\,Q\,z \Rightarrow y <> z$ (resp. $\forall x,\ y\,P\,x$ et $z\,Q\,x \Rightarrow y <> z$)

+ P et Q sont **incompatibles-objet** (resp. **incompatibles-valeur**) ssi:

$\forall x,\ x\,P\,y \Rightarrow \neg\,(\exists z \text{ tq } x\,Q\,z)$ (resp. $\forall x,\ y\,P\,x \Rightarrow \neg\,(\exists z \text{ tq } z\,Q\,x)$)

On écrira ces relations dans la MBF, respectivement:

P **DifférentObjet** Q
P **DifférentValeur** Q
P **IncompatibleObjet** Q
P **IncompatibleValeur** Q

Exemples:

1) Si on regarde le système de prévision de vente donné en exemple au chapitre précédent, les deux prédicats:
Chiffre_D_Affaire_Hors_Taxe et Chiffre_D_Affaire_TTC sont différents-objet. (par ailleurs, ils sont aussi synchrones-objet)

2) Imaginons qu'on manipule des ensembles. Ces ensembles peuvent être décrits en extension ou en compréhension. On a donc deux prédicats, DefiniEnExtension et DefiniEnCompréhension qui associent à l'identificateur d'un ensemble, soit la liste de ces éléments, soit la formule permettant de les reconnaître.
DefiniEnExtension et DefiniEnCompréhension sont incompatibles-objet.

Remarque:

Ces notions ne sont évidemment pertinentes que si les domaines conceptuels des prédicats concernés coïncident: il est clair en effet que c'est lorsque deux relations ont le même domaine conceptuel valeur que cela a un intérêt de déclarer que pour un même élément elles fournissent des images différentes.

Remarque générale sur la partie b.2):

Comme pour le niveau 2 de la partie précédente, on peut étendre ces définitions pour tenir compte de la suffisance contextuelle des prédicats.

Par exemple, le synchronisme-objet, dans le cas où P et Q sont de suffisance contextuelle, s'écrit:

$\forall G, \forall x,$ si $(x\ P\ y)\ G\ y'$ alors $\exists z$ tq $(x\ Q\ z)\ G\ y'$

Et $\forall G, \forall x,$ si $y'\ G\ (x\ P\ y)$ alors $\exists z$ tq $y'\ G\ (x\ Q\ z)$

c) Connaissances sur les concepts

Notre langage d'expression de connaissances sur les concepts est assez pauvre. En effet, notre approche de la métaconnaissance est centrée sur le prédicat, le concept n'étant qu'un élément de description des prédicats. Ce que l'on va présenter maintenant n'est donc que le "minimum nécessaire" pour compléter les informations fournies dans la partie précédente. Il n'est donc pas question d'introduire ici de nouvelles sortes decontraintes. A titre d'exemple, une contrainte du type: "La moyenne des salaires des représentants est égale à 10000 francs", appelée "contrainte sur données groupées" dans le cadre des bases de données, et qui pourrait facilement s'appliquer à des concepts, ne nous intéressent pas ici.

c.1) Propriété intrinsèque

La seule propriété intrinsèque à un concept que nous donner au système est sa définition en extension. On peut en effet, grâce au prédicat **DefiniPar,** associer **une liste d'objets** Genesia **à un concept**. On écrira alors dans la MBF, pour un concept c:

c **DéfiniPar** L,
avec L un triplet de la forme (x1 LISTE (x2 LISTE (x3 ... (x_n LISTE NIL) ...)))

Exemples:

1) On reprend l'exemple du b.2.1). On a vu alors le concept un_taux qui représentait l'ensemble {"+", "-", 0}. On écrit donc:
un_taux DefiniPar ("+"Liste ("-"Liste (0 Liste NIL)))

2) Ou, par exemple, le prédicat EN_ETAT_DE_MARCHE signalant si un matériel est en état de fonctionner ou non sera décrit comme ayant en domaine conceptuel valeur le concept une_valeur_booleenne., lui-même décrit par la relation:
une_valeur_booleenne DefiniPar (VRAI Liste (FAUX Liste NIL)).

Les seules valeurs constantes autorisées à des places, dans un triplet, réservées à des instances d'un concept défini en extension, seront bien sûr celles correspondant à l'ensemble fourni dans la MBF.

Remarques:

1) Pour être complet, ajoutons qu'il est possible de déclarer seulement que le concept est défini par un ensemble statique:
une_mention DefiniPar un_ensemble_statique
C'est alors le système qui cherchera, en examinant la BdR, à construire la liste d'objets GENESIA définissant ce concept. (voir 2)c)).

2) Bien que cela n'est pas été abordé dans VAUBAN1, il nous semble intéressant de signaler qu'il serait sans doute utile, pour affiner le concept de contradiction ou mieux contrôler les phénomènes de non-monotonie de classer les ensembles de valeurs autorisées d'une relation.

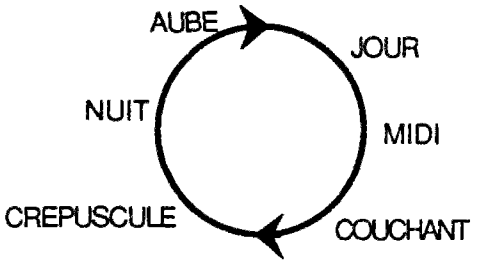
On pourrait, en s'inspirant de la linguistique, distinguer:

- les ensembles composées de termes **antinomiques** (contraires).

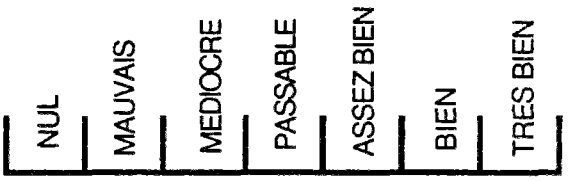
Par exemple, la liste (VRAI Liste (FAUX Liste NIL))

- les ensembles (ordonnés) contenant les valeurs successives d'un phénomène circulaire (périodique): on parle alors de **circularité**.

- les ensembles (ordonnés) contenant les valeurs successives d'un phénomène linéaire. on parle alors de **linéarité**.



Circularité



Linéarité

- les ensembles (ordonnés) de valeurs suivant une progression sur plusieurs critères successifs. Par exemple, notons "a" le fait d'avoir la propriété a et ¬a le fait de ne pas l'avoir, on peut avoir la **progression**: (tabouret, chaise, fauteuil, canapé):

a = /pour s'asseoir/	a ¬b ¬c ¬d = tabouret
b = /avoir un dossier/	a b ¬c ¬d = chaise
c = /avoir des bras/	a b c ¬d = fauteuil
d = /pour plusieurs personnes/	a b c d = canapé

c.2) Relations entre concepts

Les relations entre concepts sont de deux ordres. Celles établissant un lien entre deux concepts et celles permettant de définir un troisième concept comme le résultat d'une opération sur deux autres concepts.

La première catégorie ne contient pour l'instant que la relation d'inclusion entre concept. Ainsi on peut écrire dans la MBF, entre deux concepts c1 et c2 que:

c1 **InclusDans** c2

pour signaler que **toute instance du concept c1 est également une instance du concept c2.**

Exemples:

1) une_pompe InclusDans un_materiel

nous indique que tout ce qui concerne le concept un_materiel est également valable pour le concept une_pompe.

2) une_union_de_concepts InclusDans un_concept

pour indiquer, dans notre système, que l'union de deux concepts est un concept.

La deuxième catégorie contient les relations suivantes:

+ L'union binaire entre concepts, signalée dans la MBF de la façon suivante:

(c1 **UnionBinaire** c2) **DefinitLeConcept** c3

pour indiquer que le concept c3 est la réunion de c1 et de c2

+ La définition d'un concept comme ensemble des instances d'une relation ou d'un ensemble de relations (voir a.1)) , signalé dans la MBF sous la forme, respectivement:

((c1 **AssocieA** c2) **ParLaRelation** R) **DefinitLeConcept** c3 et ((c1 **AssocieA** c2) **ParToutesLesRelationsSemantiquementPossibles** OUI) **DefinitLeConcept** c3

avec c1, c2, c3 des concepts, et qui indique que le concept c3 est respectivement:

1er cas : le représentant de l'ensemble des relations par le prédicat R entre les instances du concept c1 et les instances du concept c2.

2ème cas : le représentant de l'ensemble des relations entre les instances du concept c1 et les instances du concept c2.

Note: - dans le premier cas: Si R est de suffisance contextuelle et que c3 est le domaine conceptuel objet ou valeur d'un prédicat Q , on dira que :

Q est un prédicat complémentaire de R

- dans le deuxième cas : Si R_i est un élément de l'ensemble des relations de domaines conceptuels objet c1 et valeur c2 et si R_i est de suffisance contextuelle, et que c3 est le domaine conceptuel objet ou valeur d'un prédicat Q , on dira que :

Q est un prédicat complémentaire de R_i

Exemples:

1) VAUBAN1 manipule des objets qui représentent des règles. Il contient des relations qui concernent les prémisses, d'autres les conséquents, d'autres indifféremment l'un ou l'autre. D'où, dans sa métabase la présence:

. des concepts: une_prémisse, un_conséquent, une_proposition

. de la relation:

(une_prémisse UnionBinaire un_conséquent) DefinitLeConcept une_proposition

2) On reprend l'exemple du b.1.1). On a vu que un plan enchaînait des séquences d'actions, une séquence d'actions étant définie par deux actions reliées par le prédicat SUIT. On écrit donc:

(une_action AssocieA une_action) ParLaRelation SUIT) DefinitLeConcept une_séquence_d_actions

3) Le système SIREN (déjà cité) manipule des contraintes. Une contrainte est une expression du type V1 > V2 - V3. Il a donc 4 prédicats représentant les 4 opérations arithmétiques : "+", "-", "/", "*" et 3 prédicats représentant les relations ">", "<" et "=". Leur description (partielle) est:

- "+" DCO[13] une_variable,
  "+" DCV[14] une_variable_ou_un_nombre
  et de même pour "-", "/" et "*".

---

[13] DomaineConceptuelObjet

[14] DomaineConceptuelValeur

- on définit le concept d'expression arithmétique élémentaire par la relation:
((une_variable AssocieA une_variable_ou_un_nombre)
ParToutesLesRelationsSemantiquementPossibles OUI)
DefinitLeConcept une_expression_arithmétique_élémentaire
- on écrit:
">" DCO une_expression_arithmétique_élémentaire
">" DCV une_expression_arithmétique_élémentaire
et de même pour "<" et "=".
- on peut alors définir le concept de contrainte:
((une_expression_arithmétique_élémentaire AssocieA
une_expression_arithmétique_élémentaire)
ParToutesLesRelationsSemantiquementPossibles OUI)
DefinitLeConcept une_contrainte

Remarques:

1) Comme nous l'avons déjà dit pour la relation de complémentarité entre prédicats, il faudrait mieux définir l'union de concepts comme un opérateur n-aire, même si les notations sont alors un peu plus lourdes (cf [DUCLOS90]).

2) Le cas de définition d'un concept c par la métarelation:
((c1 AssocieA c2) ParToutesLesRelationsSemantiquementPossibles OUI)
DefinitLeConcept c n'est pas qu'une facilité d'écriture.
Il est vrai que si tous les prédicats sont décrits dans la MBF, le concepteur peut recenser toutes les relations dont les instances doivent être représentées par le concept c, et le définir comme tel. La métarelation ci-dessus est alors inutile. Par contre, si tous les prédicats ne sont pas décrits dans la MBF, les deux métarelations ne sont plus équivalentes. En effet, supposons par exemple:
- que R1 ne soit pas décrit dans la MBF et que:
- qu'on sache grâce à la MBF que le domaine conceptuel objet de R2 est c,
- qu'on ait dans la BdR la proposition : ((x1) R1 (y1)) R2 (z1)
Alors, si on a:
... ParToutesLesRelationsSemantiquementPossibles OUI) DefinitLeConcept c
aucune erreur ne sera détectée. La proposition sera utilisé pour construire des domaines conceptuels potentiels objet et valeur de R1.
Par contre, si on a décrit c comme le représentant des instances d'un ensemble figé de relations, et que R1 n'appartient pas à cet ensemble, une erreur sera générée.

3) Il est clair que les métarelations :
... AssocieA ... ParLaRelation ... et
... AssocieA ... ParToutesLesRelationsSemantiquementPossibles OUI)
sont celles qui **répondent à la déclaration, pour un prédicat, de Suffisance CONTEXTUELLE.**
Plus exactement, si un prédicat R, de domaines conceptuels objet et valeur c1 et c2, est déclaré de suffisance contextuelle, on doit:
- avoir une métarelation:
((c1 AssocieA c2) ParLaRelation R) DefinitLeConcept ...
ou une métarelation:

((c1 AssocieA c2) ParToutesLesRelationsSemantiquementPossibles OUI)
DefinitLeConcept ...
permettant de définir un concept c3

- que le concept ainsi défini soit le (une partie du) domaine conceptuel objet ou valeur d'un autre prédicat.

4) Notons que la relation DefinitLeConcept est bijective. Une association définit un concept, et un concept est définit par une association.
Par exemple, si l'on veut que le prédicat P est en place objet des triplets de la forme (. Q1 .) ou (. Q2 .), il faut opérer en trois temps:

1°) définir le concept "(. Q1 .)"
2°) définir le concept "(. Q2 .)"
3°) définir le concept résultat de l'union binaire des deux concepts ci-dessus, et l'indiquer comme domaine conceptuel objet de P.

On ne peut pas en effet écrire:
((... AssocieA ...) ParLaRelation Q1) DefinitLeConcept c
((... AssocieA ...) ParLaRelation Q2) DefinitLeConcept c
et écrire que le domaine conceptuel objet de P est c.

Remarque générale sur le c.2):

Nous n'avons pas établi de relation d'intersection. Elle pourrait s'avérer utile. On introduira alors le schéma calqué sur celui de l'union binaire:
(c1 IntersectionBinaire c2) DefinitLeConcept c3

3 - Vérification de la Cohérence d'une Base de Connaissances : Principes

a) Définitions Générales

Nous allons ici poser les définitions nécessaires à la compréhension des parties suivantes.

a.1) Notations

On reprend en partie les notations de la note [HERY&L85]. Aussi désignerons-nous: . la **base de faits initiale** par **BFi**,

. la **base de faits finale** par **BFf**,

. le processus d'inférence par **INF(BFi, BR)** ou BR désigne la base ordonnée de règles servant aux déductions, et on dira que BFf = INF(BFi, BR).

. la base de faits courante lors de l'inférence par **BF**

On notera en abrégé **P.I.N. i** , i ∈ {1, 2, 3, 4}, pour Propriétés Intrinsèque de Niveau i (propriétés des prédicats), et **P.I.C.** pour Propriétés Intrinsèques des Concepts.

a.2) Définitions sur les termes d'une règle

variable élémentaire, variable objet

On dira qu'une variable X d'une règle R est une variable élémentaire de cette règle si elle n'apparait jamais dans la règle pour représenter un triplet dans un autre triplet. Toute variable non élémentaire de la règle sera appelée variable objet.

Exemple:

REGLE r1
SI F(X) = (Y) : (Z)
G(Z) = (T) ' (U)
a R (X) : (U)
H a = (X) : (V)
ALORS
P(X) = (Y) : (W)
Q(V) = VRAI
R(V) = (Z)

Ensemble des variables de r1 = {X, Y, Z, T, U, V, W}
Sous-ensemble des variables élémentaires = {X, Y, T}
Sous-ensemble des variables objet = {Z, U, V, W}

Remarques:

Une variable objet ne sert à désigner qu'un seul triplet dans une règle. Elle n'est donc qu'une seule fois présente en position objet dans une prémisse ou un conséquent de la règle.

On n'oubliera que quand on écrit F(X) = (Y) : (Z), (Z) désigne en fait la prémisse F(X) = (Y) 'VRAI (statut implicite).

valeur en cours d'inférence d'un terme

On note $VAL_{i,r}(x)$ la "valeur" du terme x au $i^{ème}$ déclenchement de la règle r , définie de la façon suivante :

$VAL_{i,r}(x) = x$ si x est une constante

$VAL_{i,r}(x) = a$ si a est l'instanciation de la variable x pour ce déclenchement,

donc si x est une variable objet, ie un terme de la forme (z) R (t) avec z et t des variables, on a

$VAL_{i,r}(x) = VAL_{i,r}(z)$ R $VAL_{i,r}(t)$

En cas d'ambiguïté, on rajoute un identificateur de l'inférence :

$VAL_{i,r,INF1}(x)$ désigne la valeur du terme atomique x au $i^{ème}$ déclenchement de la règle r dans l'inférence INF1 (INF1 désignant une INF(BFi, BR)).

On note $=_s$ et $<>_s$ les relations d'égalité et de différence stricte sur (l'ensemble des termes Genesia d'une règle)[2]. Ces relations sont définies par:

- **$x =_s y$** si x et y sont deux termes d'une règle r et que **$\forall i$, $VALi,r(x) = VALj,r(y)$**

ie si x et y sont des constantes identiques, ou des variables ayant des instanciations forcément égales, ou si x (resp. y) est une variable instanciable uniquement par la constante y (resp. x).

- **$x <>_s y$** si x et y sont deux termes d'une règle r et que**$\forall i$, $VALi,r(x) <> VALj,r(y)$**

ie si x et y sont des constantes différentes, ou des variables ayant des instanciations forcément différentes, ou si x (resp. y) est une variable jamais instanciable par la constante y (resp. x).

Remarques:

1) On n'a pas $x <>_s y = \neg (x =_s y)$

2) les relations sont symétriques: par exemple $x =_s y \Rightarrow y =_s x$

a.3) Définitions sur les règles

On notera **PREM(r)** l'ensemble des prémisses de r vu comme une seule formule logique et **CONS(r)** l'ensemble de ces conséquents vu comme une seule formule logique.

La règle r pourra donc s'écrire :

règle r : SI PREM(r) ALORS CONS(r)

Satisfiabilité d'une règle dans un ensemble de faits

Une règle r est satisfiable dans un ensemble de faits EF s'il existe une instanciation globale des variables de r dans EF.

Exemple :

EF = {a R1 b, a R2 c, b R3 c, d R1 e, ...}

règle r : SI (X) R1 (Y)
(X) R2 (Z)
(Y) R3 (Z)
ALORS (Z) R4 (X)

La règle est satisfiable dans EF, l'instanciation globale de ces variables étant:

(X) instancié par a, (Y) instancié par b et (Z) instancié par c

a.4) Définitions et notations liées à l'inférence

numéro de déclenchement, instanciation globale, faits associés

On associe au déclenchement d'une règle r le numéro n si ce déclenchement est le $n^{ième}$ depuis le début de l'inférence.
L'instanciation globale correspondante sera notée **IG(n,r)**, et l'ensemble des prémisses instanciées qui sont des faits de la base de faits courante, ie les faits associés à ce déclenchement **FA(n,r)**.

Exemple:
si le $3^{ème}$ déclenchement de l'inférence correspond au déclenchement de la règle r

SI F a = b
G(X) = a
(X) > 0
ALORS H(X) = b
avec (X) instancié par 1,

On aura IG(3, r) = {((X), 1)}
et FA(3, r) = {a F b, 1 G a}

b) Vérification de Propriétés et Cohérence

Nous avons défini un langage d'expression de métaconnaissances. Nous allons montrer, dans les parties suivantes, comment confronter la base de règles et cette métaconnaissance. Ce travail s'apparente à celui de la vérification de la cohérence d'une base de connaissances. Nous allons rapidement montrer pourquoi, et situer notre approche, en nous appuyant sur les définitions posées dans l'ouvrage de synthèse: "La cohérence dans les bases de connaissances", écrits par Marc AYEL et Marie-Christine ROUSSET ([AYEL&R90]).

Les contraintes que nous avons définies se rapprochent beaucoup de celles définies par M. Ayel et M.C. Rousset. Elles sont un peu moins générales dans la mesure où nous n'avons pas de langage complet d'expression de contraintes d'intégrité, un peu plus générales dans la mesure où nous permettons l'expression de contraintes de déclenchabilité systématique de règles (voir la remarque du 3)b)b.1)b.1.2)), mais globalement, les ressemblances sont très importantes. Nous pouvons donc dire que nous avons défini dans le 2) ce que ces auteurs appellent un C-Modèle. On définit alors la cohérence de la base de règles de la façon suivante: une base de connaissances est cohérente si à partir d'une base de faits initiale conforme au C-Modèle, la base de règles génère des bases de faits courantes (et enfin une base de faits finale) conformes au C-Modèle. Nos intentions sont identiques. Nous parlerons donc aussi de **cohérence d'une base de connaissances**.

Par ailleurs, comme ces deux auteurs l'ont montré, on distingue deux approches de la vérification de la cohérence, et pour chacune de ces approches, deux méthodes. Ce sont:

- l'approche de la vérification de la cohérence par une série d'examens **locaux**, ie ne s'intéressant qu'à quelques faits ou quelques règles à la fois (**cohérence statique**), ou l'approche par un examen **global** de la base de règles, ie en s'intéressant à toutes les possibilités d'interaction entre règles (**cohérence dynamique**).

Parfois, ces deux approches (statique ou dynamique) servent pour prouver les mêmes propriétés: par exemple, le système LRC3 de [HERY&L85] prouve statiquement la conformité au même C-Modèle que MELODIA ([CHARLES90]) qui, lui, apporte une preuve dynamique. La différence est qu'avec l'approche statique, la vérification est plus brutale, et ne permet que d'établir une condition suffisante de respect de la cohérence de la base de règles.

Avec d'autres C-modèles, par contre, l'approche statique s'avère insuffisante pour assurer la vérification de certaines propriétés. Ainsi, dans SACCO ([AYEL&al88]), certaines propriétés du modèle, prouvables statiquement, le sont effectivement, alors que les autres, que l'on ne peut appréhender que de manière dynamique, sont (tentées d'être) démontrées dynamiquement.

Illustrons cette distinction sur un exemple. Soit la règle:

SI (X) F a et (X) G b ALORS (X) H c .

Supposons que l'on sache seulement que:

- les valeurs autorisées pour F en place objet sont l'ensemble des nombres multiples de 2
- les valeurs autorisées pour G en place objet sont l'ensemble {4, 8}
- les valeurs autorisées pour H en place objet sont l'ensemble des diviseurs de 12

Sur cette règle et avec cette métaconnaissance, il convient de vérifier:

- que les valeurs autorisées en place objet pour F, G et H sont compatibles, ie qu'il existe au moins une instanciation de X qui satisfasse les prémisses de la règle sans être contradictoire avec la déduction de "(X) H c". Ici, *en examinant seulement la règle et les contraintes,* on trouve que X, instancié par la valeur 4, satisfait les contraintes associées à F, G ET H.
- que X ne peut s'instancier par une valeur qui amènerait cette règle à générer des faits contradictoires. C'est-à-dire de vérifier que X ne peut s'instancier par 8. En effet, cette valeur satisfait les deux contraintes associées aux prédicats présents en partie prémisse, mais viole celle associée à H, générant le fait contradictoire 8 H c.

Or, même si on ne peut pas conclure en regardant la règle seule, il serait stupide de déclarer que la base de règles est incohérente: il faut ici chercher, *en examinant les contextes de déductions de F et G,* si les deux faits 8 F a et 8 G b peuvent ou non coexister.

On a donc ici deux vérifications compléméntaires, l'une effectuable de façon statique, la seconde, *par sa nature,* à effectuer de manière dynamique.

- pour chacune de ces approches, on peut procéder de manière **systématique**, ie procéder à l'examen tous les cas problématiques (**approche exhaustive**), ou procéder de manière **approximative**, en sélectionnant par des heuristiques les examens qui "méritent" d'être faits (**approche heuristique**).

Dans les faits, on constate que:

- la majorité des systèmes travaillant sur des bases de connaissances d'ordre Zéro (Plus) adoptent une approche "cohérence dynamique" de la vérification, et procèdent de manière **exhaustive**: cf COVADIS ([ROUSSET88a], [ROUSSET88b]), INDE ([PIPARD87], [PIPARD88]), SUPER ([FONT&LEB86], [FONT&al88]) , MELODIA (CHARLES90], CHAR&D90]), JASMIN ([BEAUVI88], [BEAUVI89]).
Nous avons d'ailleurs fait de même avec MELOMIDIA.

- pour les systèmes travaillant sur des bases de connaissances d'ordre Un, mis-à-part COCO ([LOISEAU89], [LOISEAU90]), à mon avis inutilisable sur de "vraies" bases de règles d'ordre Un, l'approche est mixte. D'une part, les vérifications statiques sont effectuées de manière exhaustive, les vérifications dynamiques étant pour leur part effectuées de manière **heuristique**: cf SACCO ([AYEL87], [AYEL88], [AYEL&al89]), TIBRE ([LALO88], [LALO89]).

La conclusion de cette étude est donc que, lorsque la logique n'est pas trop compliquée, il faut être ambitieux et garantir le contrôle "maximal", mais que lorsqu'on augmente la puissance d'expression du langage, on doit effectuer statiquement ce qui peut l'être, et, pour le reste, se résoudre à perdre l'exhaustivité.

Dans notre cas, et si l'on se met de coté les propriétés intrinsèques au prédicat de niveaux 3 et 4, on s'aperçoit qu'une approche purement statique, étant donnée la richesse du métalangage, couvre à elle seule un large spectre de contrôles. Par ailleurs, elle est compatible, en terme de faisabilité, avec la puissance d'expression de Genesia2. Nous avons donc retenu cette approche, et nous présenterons donc dans les paragraphes suivants une méthode exhaustive de vérification de la cohérence statique d'une base de règles. Toutefois, il est clair que ces vérifications ne suffisent pas à garantir totalement le fait qu'aucun fait contradictoire ne soit généré lors d'un processus d'inférence: certains contrôles, dans notre C-modèle, ne pouvent, comme dans SACCO, se concevoir que dynamiquement (nous préciserons lesquels au cours de ce chapitre, et dans sa conclusion). Mais nous verrons surtout que le travail effectué lors de la vérification statique a, avec notre métalangage (il a été conçu pour), une portée importante.

c) Cohérence, Contradiction et Conformité

Nous allons ici préciser, tant en ce qui concerne les faits qu'en ce qui concerne les règles, ce que recouvre la notion du respect du modèle donné dans la MBF.

c.1) Contradiction dans les Faits

Un fait ou un ensemble de faits est contradictoire s'il ne satisfait pas le modèle décrit dans la métabase. Au vu des propriétés dont est constitué le modèle, on peut donc rencontrer trois sortes de contradiction. Il peut y avoir:

- **contradiction intrinsèque** (à un fait): si le fait en lui-même est contradictoire avec le modèle. Par exemple, si on écrit le fait:

POMPE-PRIMAIRE En_Marche HORS_SERVICE, alors que l'on sait, grâce à la métarelation DefiniPar, que En_Marche ne peut prendre que la valeur OUI, il y a contradiction interne.

- **contradiction par coexistence** (entre deux faits): si la présence simultanée de deux faits entraîne la violation d'une des propriétés du modèle.

Par exemple, il y a contradiction par coexistence si on a les deux faits x R y et x R z, qu'on sait que R est une fonction, et que y est différent de z.

- **contradiction par absence** (dans un ensemble de faits): si dans un ensemble de faits, la violation d'une des propriétés du modèle résulte de l'absence d'un fait dans cet ensemble.

Par exemple, si dans un ensemble de faits, on a le fait a R b sans qu'il existe le fait b R a alors que R possède la valeur SYMETRIQUE pour son champ TypePhysique, il y a contradiction par absence.

On peut donc dresser le tableau suivant:

Type de Contradiction	Nombre d'éléments en jeu	Propriétés du modèle concernées
Intrinsèque	1	P.I.N 1 P.I.C. (DefiniPar) AssociéA ParLaRelation
Par Coexistence	2	Fonction, Injection, Antisymétrie Différent, Incompatible
Par Absence	1 ensemble de faits	Recouvrement, Symétrie, Réflexivité Synchronisme, Complémentarité

On dira qu'un ensemble de faits est non-contradictoire si tous ces éléments sont intrinsèquement non-contradictoires, qu'il n'y a aucune paire de faits dont la coexistence est contradictoire, et s'il n'est pas, dans son ensemble, contradictoire par absence.

c.2) Contradiction dans les Règles

Les propriétés énoncées dans la MBF peuvent se séparer en deux catégories. Il y a, d'une part, les propriétés relatives aux faits manipulées par le système, ie celles présentes dans le tableau ci-dessus, et d'autre part celles faisant référence à une sémantique plus profonde du prédicat, et auxquelles correspondent des contraintes sur les règles ou sur la base de règles prise dans son intégralité. Cette deuxième catégorie est constituée des métafaits données aux niveaux 3 et 4 (niveau mathématique et niveau fonctionnel) des propriétés intrinsèques aux prédicats.

Il est donc naturel de distinguer **deux sortes de contradictions dans les règles:**

- celles correspondant au fait de pouvoir engendrer des faits contradictoires lors de l'inférence,

- celles correspondant à la violation de la fonction d'un prédicat (niveau 4) dans la BdR, ou la violation de sa nature mathématique (niveau 3: plus

précisément, il s'agit de la violation des obligations nées de la différence entre la nature mathématique et la nature physique du prédicat).
Nous ne nous intéresserons qu'au premier type de contradictions. Le deuxième mériterait sans doute à lui tout seul une thèse. Il nous a semblé hors de notre portée dans le cadre de ce travail, quoique envisageable à moyen terme.

Nous distinguerons **trois cas** dans le premier type de contradictions. On parlera:

- de règle intrinsèquement contradictoire: une règle R est intrinsèquement contradictoire s'il existe un ensemble non contradictoire de faits EF tel que tout déclenchement de R avec une instanciation de ces variables dans EF conduise à la déduction d'au moins un fait :

- intrinsèquement contradictoire : la règle est alors dite intrinsèquement contradictoire de type 1

ou - contradictoire par coexistence avec l'un des faits instanciant les prémisses de la règle : la règle est alors dite intrinsèquement contradictoire de type 2

Exemple:
```
REGLE r1
SI          P(X) = OUI
ALORS       Q(X) = VRAI
REGLE r2
SI          F(X) = a
ALORS       F(X) = b
```
avec
```
Q DomaineConceptuelValeur un_état
un_état DefiniPar (OUVERT Liste (FERME Liste NIL))
F TypePhysique FONCTIONNEL
```
Tout déclenchement de r1 crée un fait intrinsèquement contradictoire
donc r1 est intrinsèquement contradictoire de type 1
Tout déclenchement de r2 crée un fait contradictoire par coexistence avec le fait instanciant la prémisse, donc r2 est intrinsèquement contradictoire de type 2

- de couple contradictoire de règles : le couple de règles (reg1, reg2) est contradictoire s'il existe un ensemble de faits EF non contradictoire tel que si on pose: EFf1 = INF(EF, {reg1}), EFf2 = INF(EF, {reg2}), alors EF' = EFf1 $\cup$ EFf2 contient un couple de faits contradictoire par coexistence.

Exemple:
```
REGLE r1    SI          R(X) = a
            ALORS       F(X) = a
REGLE r2    SI          R(X) = b
            ALORS       G(X) = b
```
avec en MBF la seule métarelation: F IncompatibleValeur G. On pose EF = {a R a, a R b}, on a EFf1 $\cup$ EFf2 = {a F a, a G b} est contradictoire par coexistence.

- d'un ensemble contradictoire de règles : l'ensemble {reg1, reg2, ... , regN} de règles est contradictoire s'il existe un ensemble non contradictoire de faits EF tq si: EFf1 = INF(EF, {reg1}), EFf2 = INF(EF, {reg2}), ... , EFfN = INF(EF, {regN}), alors

$EF' = \bigcup_{i \in \{1, ..., N\}} INF(EF, \{reg_i\})$ est contradictoire par absence

Exemple : Si une base de règles BR1 a pour seule règle concluant sur S3 la règle

règle reg1 :

SI (X) S1 (Y)
(Y) S2 (Z)
ALORS (X) S3 (Z)

avec S1, S2 et S3 déclarés de type physique symétrique, il est facile de montrer que tout ensemble de règles de BR1 contenant reg1 est contradictoire, le prédicat S3 provoquant des contradictions par absence dans les BF générées.

Remarques:

1) Dans le cas d'un couple contradictoire de règles, les règles ne sont pas nécessairement distinctes. On peut très bien par exemple avoir le couple (r1, r1) qui soit contradictoire. Dans ce cas, la réunion des ensembles issus des inférences se réduit à INF(EF,{r1}). La notion importante ici est la nécessité d'exécution de plusieurs déductions pour avoir contradiction.

2) Un ensemble de règles peut, bien sur, générer aussi des couples de faits contradictoires par coexistence: l'ensemble EF' ci-dessus peut très bien en contenir. Toutefois, puisqu'alors la contradiction s'exprime par l'intermédiaire de deux faits déduits, cette contradiction provient de deux conséquents, et est donc produite par un couple de règles. On ne s'intéressera donc pas aux contradictions par coexistence engendrées par un n-uplet de règles.

c.3) Vérification des métarelations, Contradiction dans les Règles et dans les Faits

Nous allons par la suite vérifier la base de connaissances par une série d'examens, chaque examen correspondant à la vérification du respect d'un ou de plusieurs types de métarelations. Donnons ici à quel(s) type(s) de contradictions dans les règles et les faits la vérification d'une métarelation correspond:

Métarelation de la MBF	Type de Contradiction dans les Règles	Type de Contradiction dans les Faits
P.I.N. 1 P.I.C. (DefiniPar) AssocieA	intrinsèque type 1	intrinsèque
TypePhysique FONCTIONNEL INJECTIF ANTISYMETRIQUE	intrinsèque type 2 couples de règles	par coexistence
TypePhysique SYMETRIQUE REFLEXIF SYNCHRONE COMPLEMENTAIRE	n-uplet	par absence
INCOMPATIBLE DIFFERENT	intrinsèque type 2 couples de règles	par coexistence

c.4) Validité d'une Base de Faits Initiale, Conformité, Cohérence

Les vérifications des propriétés énoncées dans la MBF s'appuient sur les propriétés énoncées dans la MBF. Autrement dit, nous aurons souvent des démonstrations du type: P et Q sont incompatibles-objet si F et G sont de type physique injectif. Pour pouvoir conclure sur la non-violation de ses propriétés lors de l'inférence, nous aurons donc besoin de connaître les propriétés "sûres", ie non contestables en examinant la base de règles. Par exemple, ci-dessus, si F et G n'apparaissent jamais en partie conséquent dans la base de règles, alors, sous réserve de contrôler les BFi, on peut dire qu'on est sûr que F et G sont de type physique injectif. Du coup, on peut alors affirmer qu'on est sûr que P et Q sont incompatibles-objet. La conformité d'un prédicat est une formalisation de cette notion de "sûreté" d'un prédicat pour une propriété donnée.

On va préciser quelles sont, dans notre système, les bases de faits initiales valides pour l'inférence, puis donner les définitions relatives à la conformité d'un prédicat.

Validité d'une base de faits initiale

Une **base de faits initiale valide est un ensemble non contradictoire de faits, chaque fait étant une instance d'un prédicat non déductible.**

Remarque: Cette définition est la correspondante en logique d'ordre Un de celle donnée en logique d'ordre Zéro Plus dans la partie 1.

Conformité d'un prédicat à une de ses propriétés, propriété opérante

Un prédicat est conforme à une de ses propriétés si, à partir de toute base de faits initiale valide, on ne peut déduire au cours de l'inférence d'instance de ce prédicat qui soit contraire avec la propriété en question. On dira alors que la propriété est opérante sur ce prédicat.

Conformité d'un prédicat à un de ses champs sémantiques

Plus généralement, on dira qu'un prédicat est conforme à son champ sémantique C si, à partir de toute base de faits initiale valide, on ne peut déduire, au cours de l'inférence, d'instance de ce prédicat qui contredise le ou les propriétés énoncées dans le champ sémantique C de ce prédicat.

Par exemple, si le prédicat P possède comme type physique le couple (FONCTION, INJECTION), il sera conforme à son champ TypePhysique si les propriétés "P TypePhysique FONCTION" et "P TypePhysique INJECTION" sont opérantes.

Conformité Totale d'un prédicat

Un prédicat est totalement conforme si, à partir de toute base de faits initiale valide, on ne peut déduire, au cours de l'inférence, d'instance de ce prédicat qui soit contraire avec une des propriétés le concernant énoncées dans la métabase.

Exception: Dans le cas où une propriété entre deux prédicats est contredite, on dira que les deux prédicats ne sont pas conformes. Donc un prédicat peut ne pas être conforme à cause de possibilité de la déduction d'instances d'un autre prédicat (en cas de synchronisme non respecté par exemple)

Remarque: Tous les prédicats non déductibles sont totalement conformes.

Etant entendu que la cohérence est le respect par la BdR de toutes les propriétés formulées dans la MBF, on a la reformulation de la cohérence suivante:

Une base de connaissances est cohérente si tous les prédicats de la base sont totalement conformes.

4 - Contrôle et Exploitation de la MBF seule, Conformité des Prédicats à leurs P.I.N. 1

Ces premières opérations visent à s'assurer:

- que l'ensemble des propriétés indiquées dans la métabase est intrinsèquement cohérent, et à le mettre en forme pour les opérations suivantes (paragraphe a),
- que les prédicats sont conformes à leurs P.I.N. 1 et lesq concepts à leur P.I.C. (paragraphes c et d). Il ne s'agira ici néanmoins que de la vérification d'une condition nécessaire de conformité. On s'intéressera ici à garantir la non-contradiction intrinsèque des règles. Cette vérification permet également de détecter un cas d'indéclenchabilité de la règle.

Enfin, nous verrons que notre système fonctionne ici tant en chaînage arrière (vérification de propriétés) qu'en "chaînage avant" (déduction de propriétés pour les prédicats nouveaux). Nous justifierons au e) cette approche. Etant donnée leur simplicité, tous ces traitements ne seront que brièvement développés.

a) Opérations sur la métabase

Les opérations effectuées ici, à partir de la seule métabase, visent trois objectifs:

a.1) Contrôler la cohérence de la Métabase

Certaines informations peuvent en effet être contradictoires avec d'autres. On effectue ici les contrôles de non-contradiction entre métafaits.
Par exemple, on va contrôler que si un prédicat est déclaré comme ayant un type physique symétrique, ses domaines conceptuels objet et valeur sont égaux. Ou que si deux prédicats sont synchrones-valeur, leurs domaines conceptuels-valeur sont égaux.

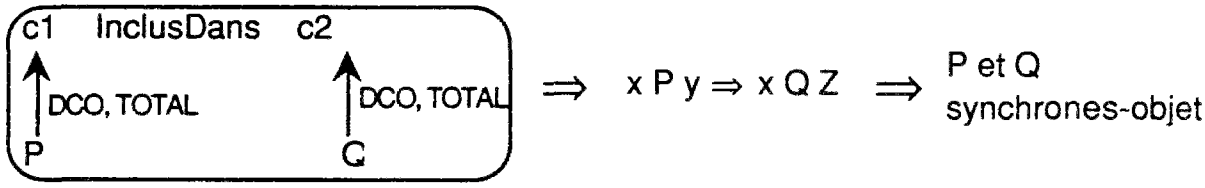
a.2) Garantir la "complètude" de la description

Dans un premier temps, certaines informations en nécessitant obligatoirement d'autres, on vérifie la présence de ces dernières.

Par exemple, on vérifie que si est prédicat est de Suffisance Contextuelle, alors le triplet qu'il définit correspond bien à un concept et que ce concept est bien le domaine conceptuel (objet ou valeur) d'un autre prédicat avec un recouvrement total, ou de plusieurs avec des recouvrements quelconques.

La deuxième partie du même module est chargée de gérer la redondance entre certaines parties de notre modèle. On déduit donc certains métafaits de ceux présents, et cela pour unifier la présentation en vue des utilisations futures.

Par exemple, si deux concepts sont définis en extension par des listes incluses l'une dans l'autre, le premier concept est inclus dans le second. Ou encore, si un concept c1 est inclus dans un concept c2, et qu'il existe deux prédicats P et Q de Domaines Conceptuels Objet (DCO) respectifs c1 et c2 et de RecouvrementObjet Total, alors P et Q sont synchrones objets:



a.3) Hiérarchiser l'ensemble des concepts

Il s'agit ici de reformuler la connaissance et de la présenter à l'utilisateur pour examen. Plus précisément on cherche les sous-concepts d'un concept donné:

- on dit qu'un concept c1 est multidimensionnel s'il existe deux relations R1 et R2 de domaines conceptuels (objet ou valeur) c1 et de recouvrement total de ce concept. Les deux concepts correspondant aux deux autres domaines conceptuels de R1 et R2 sont des sous-concepts (ou des dimensions) de c1.
- s'il existe une relation R de domaines conceptuels objet et valeur c1 et c2 et de nature fonctionnelle "attribut", c2 est un sous-concept de c1.

On présente cette hiérarchie sous la forme d'un graphe, en rajoutant les liens d'union entre concepts. L'utilisateur a ainsi sous les yeux l'ensemble des liens entre concepts, ie une vision duale de son univers de départ (il n'y a ici plus de prédicat, mais seulement que les concepts).

On donne ci-après des graphes partiels obtenus dans deux systèmes testés, SACSO (conduite de centrale) et VAUBAN1 (le notre, cette partie concerne une ancienne version du module de traitement des P.I.N 2).

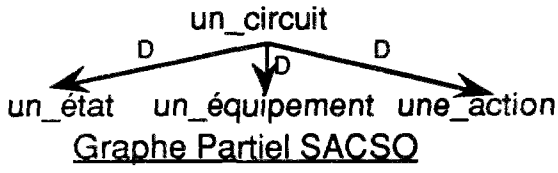
Voici la légende de ces graphes :

D indique un lien Dimensionnel entre concepts,

Att un lien Attributif (la différence avec un lien attributif est que tout instance du concept du haut n'est pas forcément en relation avec un instance du concept du bas),

U étiquetant un ovale indique une union entre les concepts contenu dans l'ovale, le concept ainsi défini étant à la frontière extérieure de l'ovale

Ass étiquetant un carré indique une association entre les deux concepts contenu dans le carré le concept ainsi défini étant à la frontière extérieure du carré.



Graphe Partiel SACSO

Graphe Partiel VAUBAN1

b) Vérification et Complémentation de P.I.C.

On va ici vérifier (et compléter) les propriétés données grâce à la métarelation DéfiniPar (suivi d'une liste d'objets GENESIA ou du mot-clé "un_ensemble_statique") et qui permet, pour un concept C de le déclarer comme représentant :
c1: un ensemble fini d'objets donné en extension
c2: un ensemble fini d'objets non fourni
sachant que par défaut, un concept représente un ensemble dépendant de la BFi et/ou de l'inférence (cas c3).

Le module CONCEPT1 vérifie une condition nécessaire de validité de cette information. Plus précisément, si un prédicat R a pour domaine conceptuel objet (resp. valeur) un concept C, alors pour toute occurrence de ce prédicat dans la BdR, ie pour chaque expression de la forme x R y en prémisse ou en conséquent:

* si x (resp. y) est une constante ,

- et que C est dans le cas c1, on vérifie que x (resp. y) appartient à la liste associée à C
- et que C est dans le cas c2, on ajoute x (resp. y) à la liste (en construction) associée à C. Cette liste sera validée par l'utilisateur en fin de traitement.
- et que C est dans le cas c3, on vérifie qu'il existe une autre occurrence de C ou le terme qui le représente est une variable . Sinon C est statique dans les règles, et doit probablement être déclaré comme tel dans la métabase.

* si x (resp. y) est une variable ,

et que C est dans le cas c1, on vérifie que la liste associée à C ne se réduit pas à un singleton (sinon l'utilisation d'une variable est inutile)[15].

et que C est dans le cas c2, on regarde si x (resp. y) ne représente pas dans la même règle un concept défini en extension. Si c'est le cas, on proposera ces valeurs comme valeurs possibles dans l'ensemble associé à C.

---

[15] Pour que la vérification soit suffisante, il faudrait ici contrôler que toutes les instances possibles pour x appartient à la liste associée à c1, ce qui relève d'une recherche dynamique (cf SACCO).

On a donc un mécanisme mixte de vérification et de complémentation de ces propriétés.

c) Vérification & Découverte des Domaines Conceptuels des prédicats

Cette partie s'occupe de vérifier la validité des jointures établies entre prédicats dans les règles, puis d'utiliser ces liaisons pour déduire de nouvelles P.I.N 1. Le traitement consiste à:

1°) 1ère PARTIE: S'assurer que les intersections des domaines conceptuels associés aux occurrences d'une même variable dans une règle sont non-vides.

Exemple: Sur la règle:

REGLE r1
SI P1(X) = (Y)
P2(X) = (Z) : (T)
P3(T) = OUI
ALORS P4(X) = OUI : (U)
P5(Y) = (U)

On procède aux vérifications suivantes:

1°) En utilisant les jointures créées par l'intermédiaire de (X)

V1.1: DCO(P1) $\cap$ DCO(P2) $\neq \varnothing$?

V1.2: DCO(P1) $\cap$ DCO(P4) $\neq \varnothing$?

V1.3: DCO(P2) $\cap$ DCO(P4) $\neq \varnothing$?

V1.4: DCO(P1) $\cap$ DCO(P2) $\cap$ DCO(P4) $\neq \varnothing$?

2°) En utilisant la jointure créée par l'intermédiaire de (Y)

V2: DCV(P1) $\cap$ DCO(P5) $\neq \varnothing$?

3°) En utilisant la jointure créée par l'intermédiaire de (T)

V3: Existe-t'il un concept c3 tel que
((DCO(P2) AssocieA DCV(P2) ParLaRelation P2) DefinitLeConcept c3
ou
((DCO(P2) AssocieA DCV(P2)) ParToutesLesRelationsSemantiquementPossibles OUI) DefinitLeConcept c3
et DCO(P3) $\cap$ c3 $\neq \varnothing$?

4°) En utilisant la jointure créée par l'intermédiaire de (U)

V4: Existe-t'il un concept c4 tel que
((DCO(P4) AssocieA DCV(P4) ParLaRelation P4) DefinitLeConcept c4
ou ((DCO(P4) AssocieA DCV(P4)) ParToutesLesRelationsSemantiquementPossibles OUI) DefinitLeConcept c4
et DCV(P5) $\cap$ c4 $\neq \varnothing$?

Ces opérations correspondent donc à la vérification :

- d'une condition nécessaire de conformité des prédicats à leurs domaines conceptuels objet et valeur[16]. Dans l'exemple ci-dessus, les tests V1.2, V1.3, V2 et V4 vérifient que les instances des relations P4 et P5 qui seront créées par cette règle ne seront pas contraires aux domaines conceptuels objet et valeur déclarés pour ce ces relations
- de la déclenchabilité de la règle, dans l'hypothèse où les prédicats sont conformes à leurs domaines conceptuels objet et valeur: test V1.1
- de la cohérence "globale" de la règle, ie la vérification que la réunion des contraintes sur les prémisses et sur les conséquents est satisfaite: test V1.4

2°) 2ème PARTIE: Découvrir, en utilisant le même type de raisonnement, la liste des Domaines Conceptuels Possibles des prédicats non décrits dans la métabase.

L'utilisateur choisit alors d'attribuer l'un des (ou une union de) domaines conceptuels possibles. On vérifie alors de nouveau la cohérence des utilisations de ce prédicat dans la BdR. En cas de cohérence, cette nouvelle connaissance peut être utilisée pour découvrir à nouveau les domaines conceptuels d'autres prédicats non-décrits.

Exemple: Soient les occurrences de propositions dans les règles suivantes:

- dans la règle r1: P1(X) = A et Q1(X) = (Y)
- dans la règle r2: P2(X) = A et Q1(X) = ...
- dans la règle r3: Q1(X) = (Y) et Q2(X) = (Y)

Supposons que la BdR ait été trouvée cohérente dans l'état courant de la métabase, qu'on connaissance les domaines conceptuels de P1 et P2 et qu'on ignore ceux de Q1 et Q2.

Le système va successivement:

1) proposer à l'utilisateur de choisir, pour DCO(Q1) entre DCO(P1), DCO(P2) et DCO(P1) ∪ DCO(P2)

2) Si l'utilisateur choisit le premier concept possible, le système vérifiera la cohérence de r2 (ie si DCO(Q1) ∩ DCO(P2) ≠ ∅), s'il choisit le second, le système vérifiera la cohérence de r1 (ie si DCO(Q1) ∩ DCO(P1) ≠ ∅)

3) Si ces vérifications sont positives, le système proposera à l'utilisateur de valider, pour DCO(Q2), la valeur DCO(Q1).

d) Vérification & Découverte du champ sémantique "Suffisance"

Enfin, le dernier volet de ces traitements concerne les prédicats de suffisance contextuelle. On va comme dans la partie précédente, effectuer la double tache de:

---

16 Pour que la vérification soit suffisante, il faudrait, comme pour les P.I.C., effectuer une recherche dynamique pour s'assurer, par exemple, que dans la règle r1, le domaine d'instanciation de X est inclus dans DCO(P4), celui de Y est inclus dans DCO(P5), et celui de U est inclus dans DCV(P5)

1°) **rechercher,** grâce aux P.I.C., **quels sont les prédicats complémentaires des prédicats de suffisance contextuelle,** et **vérifier la présence,** à l'endroit voulu, **du** (d'un des) **prédicat(s) complémentaire(s) à chacune de leurs apparitions.**

Exemple:
R1 Suffisance CONTEXTUELLE
R1 DomaineConceptuelObjet c1
R1 DomaineConceptuelValeur c2
((c1 AssocieA c2) ParToutesLesRelationsSemantiquementPossibles OUI)
DefinitLeConcept c3
c3 InclusDans c4
R2 DomaineConceptuelObjet c3
R3 DomaineConceptuelValeur c4

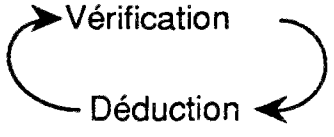
On va vérifier que chaque occurrence de R1 est de l'une ou de l'autre des formes:
1) x R1 y : z
z R2 t {: u}
2) x R1 y : z
t R3 z {: u}
avec x, y, t des termes et z {et u} des variables.

Ces vérifications reviennent donc, soit à garantir la conformité des prédicats à leur Suffisance, soit à détecter des cas d'indéclenchabilité de règle.

2°) **Proposer,** pour un prédicat non décrit dans la MBF et qui n'apparait en BdR qu'imbriqué dans d'autres prédicats, la propriété de suffisance contextuelle. Cette proposition doit être validée par l'utilisateur (on n'est pas sûr d'être en présence d'un prédicat de suffisance contextuelle).

e) Justification de la démarche, Prolongement

La "découverte" de nouvelles propriétés, quoique a priori en dehors de notre sujet, s'est avérée indispensable pour la viabilité du système. En effet, dans un système expert "moyen" (ie d'une centaine de règles), on manipule environ une centaine de prédicats (à peu près un prédicat par règle). Demander la description intégrale des cent prédicats n'est donc pas réaliste. Par contre, demander d'en décortiquer une vingtaine (en gros, ceux apparaissant en base de faits initiale) et laisser le système proposer les propriétés pour les autres est tout-à-fait pensable, et même, il me semble, souhaitable pour donner envie à l'utilisateur d'aller plus loin.

Notre système fonctionne donc ici en boucle fermée: 

Nous avions pensé également utiliser les bases de faits initiales, toujours disponibles, pour aller plus loin dans nos déductions. Le manque de temps nous a empêché de concrétiser cette idée mais nous restons persuadé que c'est une voie qui mériterait des investigations plus poussées.

5 - Conformité des Prédicats à leurs Types Physiques

Nous entrons ici dans la partie principale de notre système.

a) Fonction,Injection et Antisymétrie

a.1) Introduction

On s'occupe ici des types Fonction, Injection et Antisymétrie. On a vu (au 3)c)) que les types de contradictions sur les règles liées à ces propriétés sont:

- les contradictions intrinsèques (aux règles), de type 2:

par exemple, la règle

REGLE r2
SI F(X) = a
ALORS F(X) = b

avec F TypePhysique FONCTIONNEL
est intrinsèquement contradictoire de type 2.

- les couples contradictoires de règles :

Par exemple, le couple

REGLE r1
SI R(X) = a
ALORS F(X) = a

REGLE r2
SI R(X) = b
ALORS F(X) = b

avec en MBF la seule métarelation: F TypePhysique FONCTIONNEL
est contradictoire.

Nous verrons qu'avec nos restrictions sur la base de faits initiale, si la base de règles ne possède pas de couples contradictoires de règles, alors les règles intrinsèquement contradictoires de type 2 ne pourront jamais, dans une inférence à partir d'une BFi valide, engendrer de faits contradictoires avec ceux les instanciant (elles sont indéclenchables). On va donc concentrer notre étude sur les couples contradictoires de règles.

Cette étude va être assez longue, et assez difficile. Pour aider le lecteur, nous allons dès maintenant donner les grandes étapes de notre démarche.

Tout d'abord, nous allons, en partant de la notion de couple contradictoire de règles, introduire la notion de conflit entre règles. Deux règles seront en conflit si elles semblent pouvoir générer à elles deux des faits contradictoires par coexistence. Autrement dit, si le couple qu'elles forment est "potentiellement" contradictoire. Repérer ces conflits nécessitera, étant donné notre langage cible, d'introduire la notion d'unification entre conséquents de règles.

Puis, pour pouvoir se prononcer sur ces contradictions potentielles, nous montrerons comment construire la règle correspondant au contexte précis de déductions par ces deux règles de faits contradictoires par coexistence. Cette règle, appelée règle résolvante du conflit est une sorte de réunion des deux règles.

Nous raisonnerons ensuite sur cette règle. Nous montrerons que si on peut prouver qu'elle est indéclenchable, ou si on peut montrer que deux de ses variables (nous préciserons lesquelles) s'instancient systématiquement par la même valeur, alors la contradiction potentielle sur les deux règles est levée. On dira alors que le conflit est résolu, respectivement soit par indéclenchabilité de la règle résolvante, soit par égalité d'instance. Nous connaîtrons également quelles sont les hypothèses sur lesquelles s'appuie cette résolution. Autrement dit, quelles autres métarelations doivent être incontestablement vraies pour que le couple de règles ne soit pas contradictoire. On parlera donc de conflits résolus sous réserve d'un ensemble d'hypothèses.

Nous aboutirons alors un critère permettant de se prononcer sur la contradiction dans les couples de règles. Enfin, nous montrerons ce que ce critère permet de conclure concernant la conformité d'un prédicat à son type physique, c'est-à-dire le maintien des propriétés énoncées dans son champ sémantique "Type Physique" lors du processus d'inférence à partir d'une base de faits initiale valide.

a.2) Conflits entre règles

a.2.1) Condition Nécessaire de Contradiction entre deux Conséquents

Précisons grâce à la fonction VAL les conditions de production des contradictions envisagés ici. Il s'agit de contradiction par coexistence entre deux faits (voir 3) c.3)):

Un couple de règles (r1, r2) (avec r1 pas nécessairement distinct de r2) génère un couple de faits contradictoires par coexistence, *pour les propriétés qui nous intéressent ici*, si

1°) r1 possède un conséquent (c1) : x1 R y1

et r2 possède un conséquent (c2) : x2 R y2 , avec x1, x2, y1, y2 des termes atomiques et R une fonction ou une injection ou une antisymétrie.

2°) il existe deux ensembles non contradictoires de faits EF1 et EF2 tels que dans les deux cycles d'inférences désignées par INF1 et INF2 aboutissant à
EFf1 = INF(EF1, {r1}) et EFf2 =INF(EF2, {r2}), il existe deux déclenchements de numéro respectif i dans INF1 et j dans INF2 tels que:
si R est une fonction: $VAL_{i,r1}(x1) = VAL_{j,r2}(x2)$ **et** $VAL_{i,r1,}(y1) <> VAL_{j,r2}(y2)$
si R est une injection: $VAL_{i,r1}(y1) = VAL_{j,r2}(y2)$ **et** $VAL_{i,r1}(x1) <> VAL_{j,r2}(x2)$
si R est une antisymétrie: $VAL_{i,r1}(x1) = VAL_{j,r2}(y2)$ **et** $VAL_{i,r1}(y1) = VAL_{j,r2}(x2)$

Remarque: Pour que le couple (r1, r2) soit contradictoire, au sens défini au 3)c.2), il faut qu'on ait en plus EF1 = EF2.

a.2.2) Unification, Algorithme de Robinson

Nous allons voir que vérifier les conditions définies aux paragraphes précédents revient à vérifier la possibilité d'unifier des termes des deux règles, comme le montre l'exemple suivant:

REGLE r1
SI R1(x1) = (y1) : (z1)
R2 (x1) = (t1) : (u1)
R3 (u1) = (w1)
(y1) R5 (w1) : (a1)
ALORS
F(z1) = a
(y1) R6 (a1) : (b1)
INJ(x1) = (b1)

REGLE r2
SI R1(x2) = (y2)
R2 (x2) = (u2)
R3(y2) = (v2) : (w2)
R4 (w2) = a
ALORS
F a = (v2)
(u2) R6 (v2) : (b2)
INJ(x2) = (b2)

avec F et INJ des prédicats de type physique respectif "fonctionnel" et "injectif".

Appliquons la CN de contradiction. A priori, ces deux règles peuvent générer des faits contradictoires par coexistence dans deux cas:

1°) $\exists i,j$ tq $VAL_{i,r1}(z1) = VAL_{j,r2}(a)$ et $VAL_{i,r1,}(a) <> VAL_{j,r2}(v2)$ (contrad. sur F)

2°) $\exists i,j$ tq $VAL_{i,r1}(b1) = VAL_{j,r2}(b2)$ et $VAL_{i,r1,}(x1) <> VAL_{j,r2}(x2)$ (" " sur INJ)

Examinons les deux cas:

- cas 1: on sait que $\forall j$, $VAL_{j,r2}(a) = a$ puisque a est un objet atomique. De plus, on sait que z1 est défini par la prémisse: R1(x1) = (y1) : (z1), donc tout fait instanciant z1 est de la forme α R1 β avec α et β des objets. Il ne peut en aucun cas être égal à a, donc: $\forall i, j$ $VAL_{i,r1}(z1) <> VAL_{j,r2}(a)$.
On vient de tenter d'unifier (z1) = (x1) R1 (y1) et a, et de constater que c'était impossible.

- cas 2: b1 est défini par (y1) R6 (a1) : (b1)
avec (y1) R5 (w1) : (a1)
ie, en écriture condensée (y1) R6 ((y1) R5 (w1)) : (b1)
b2 est défini par (u2) R6 (v2) : (b2)

On voudrait savoir s'il existe une possibilité d'instanciation identique pour b1 et b2 ($VAL_{i,r1}(b1) = VAL_{j,r2}(b2)$). C'est-à-dire s'il existe une valeur d'instanciation pour chacune des variables présentes dans les deux dernières expressions de telle sorte que ces expressions instanciées soient égales.
C'est-à-dire s'il existe un semi-unificateur de ces des deux termes.

L'existence d'un semi-unificateur entre deux termes est équivalente à l'existence d'un unificateur, donc d'un **unificateur principal** entre ces deux termes, Nous chercherons donc s'il existe un unificateur principal entre les deux termes.

Rappel sur la notion d'unificateur et d'unificateur principal

* Une **substitution** est une application d'un ensemble de variables X d'un langage de signature Σ vers l'ensemble $T_\Sigma[X]$ des termes de ce langage:

$s : X \dashrightarrow T_\Sigma[X]$, telle que $D(s) = \{x; s(x) <> x\}$ soit fini.

On peut donc définir une substitution par un ensemble fini de couples (x, s(x)).

notation: **On notera également s le prolongement de s sur l'ensemble des termes**, ie l'application $\underline{s}$ définie inductivement par :

$\underline{s}(c) = c$ si c est une constante appartenant à Σ, $\underline{s}(x) = s(x)$ si $x \in X$,

et $\underline{s}(Fx_1x_2 \ldots x_n) = F(\underline{s}(x_1)\underline{s}(x_2) \ldots \underline{s}(x_n))$

* Une substitution peut-être simple (tous les couples qui caractérisent la substitution sont indépendants) ou composée (ie la composée de plusieurs substitutions simples).

Par exemple, la substitution caractérisée par l'ensemble {(x, f(a)), (y, f(x))} est composée. On doit alors appliquer les substitutions séquentiellement.

Par exemple, s(R(x,y)) = R(f(a), f(f(a)))

* Soit $F = (A_1, \ldots, A_n)$ un ensemble fini de formules atomiques, un **unificateur** de F est une substitution (composée) **u** telle que **u(A1) = u(A2) = ... = u(An)**

* Soit $F = (A_1, \ldots, A_n)$ un ensemble fini de formules atomiques, et $\mathcal{U}$ l'ensemble des unificateurs de F, **l'unificateur principal** (ou **mgu**[17]) **de F** est la substitution (composée) σ telle que tous les autres unificateurs de F s'écrivent sous forme de sa composition avec une autre substitution: $\forall s \in \mathcal{U}, \exists s' \in \mathcal{U}$ tq $s = s' \sigma$

Application à notre problème, Algorithme de Robinson

Concrètement ici:

- l'ensemble X concerné est la réunion des variables de r1 et des variables de r2 (en supposant que celle-ci a été réécrite de façon à n'avoir aucune variable en commun avec r1)

- $T_\Sigma[X]$ est l'ensemble des termes (au sens GENESIA2) **que l'on peut construire avec les constantes et les relations présentes dans r1 ou r2** (ie la "signature" de la portion de langage visible dans r1 et r2), **et X.**

- on dispose d'une procédure efficace (l'algorithme de Robinson,cf [LALEME90] p 85-86) et facile à implémenter de calcul de l'unificateur principal de deux expressions, dont on va donner la spécification:

---

[17] pour "most general unifier"

Algorithme de ROBINSON

Soit t1 et t2 les deux expressions dont on cherche le mgu. On suppose que les variables de t2 ont des noms différents de celles de t1 (sinon on les renomme).
On désigne par V(t) l'ensemble des variables d'un terme t.
Dans tout l'algorithme, x désigne une variable élémentaire, t et $t_1, \ldots, t_n$ des termes, a et b des objets atomiques.

On gère une liste L contenant les paires de termes restant à unifier. On retranche des termes à cette liste par l'opération "-", on en ajoute par "+".

a) Fonction Normaliser(L) (mise en forme normale et détection des cas d'échec)

- échec1: si L contient la paire (x, t), alors si $x \in V(t)$ alors échec de l'unification
- échec2: si L contient la paire (a, b), alors
  si a <> b alors échec de l'unification sinon L = L - ((a, b))
- échec3a: si L contient la paire (a, t1 F t2) alors échec de l'unification
- échec3b: si L contient la paire (t1 F t2, a) alors échec de l'unification
- échec4: si L contient la paire (t1 F t2, t3 G t4), alors si F <> G alors échec de l'unification
- Inversion: si L contient la paire (t , x), alors L = L - (t , x) + (x, t)
- Décomposition: si L contient la paire (t1 F t2, t3 F t4),
  alors L = L - (t1 F t2, t3 G t4)) + (t1, t3) + (t2, t4)

b) Algorithme de Construction de l'ens. des paires constituantes du mgu de (t1, t2)

- L_0 = ((t1, t2))
- L_1 = Normaliser (L_0), i = 1
- Tant que non (arrêt ou échec de l'unification)
  Si L_i est vide, alors arrêt de l'unification. Le résultat est $\sigma = \sigma_i\ \sigma_{i-1} \ldots \sigma_1$
  Sinon [choisir un élément (x, t) de L_i . On pose σ_i = (x, t)
  L_{i+1} = Normaliser($\sigma_i(L_i - (x, t))$) et itérer]

Exemple: Prenons le cas de l'exemple donné plus haut. Il s'agit d'étudier les conséquents:

cas1:	F(z1) = a,	et	F a = (v2)
avec	R1(x1) = (y1) : (z1)		
cas2:	INJ(x1) = (b1)	et	INJ(x2) = (b2)
avec	(y1) R6 (a1) : (b1)		(u2) R6 (v2) : (b2)
et	(y1) R5 (w1) : (a1)		

Les variables de r2 étant toutes différentes des variables de r1, on va tout de suite chercher les mgu des deux couples d'expressions. On obtient que:

- cas1: tentative d'unification de z1 et a

L_0 = ((x1 R1 y1 , a)) => échec de l'unification (règle échec3a)
=> le mgu de l'ensemble n'existe pas.

- cas 2: tentative d'unification de b1 et b2

au départ L_0 = (y1 R6 (y1 R5 w1), u2 R6 v2).
Après normalisation, on a L_1 = ((y1, u2), (v2, y1 R5 w1))
et s1 = (y1, u2) et

donc L_2 = ((v2, u2 R5 w1))
donc s2 = (v2, u2 R5 w1)
d'où $L_3 = \emptyset$ et Arrêt de l'unification

Le mgu est donc caractérisé par l'ensemble de paires {(y1, u2), (v2, u2 R5 w1)}. On a bien:
σ(b1) = σ(y1 R6 (y1 R5 w1)) = σ(y1) R6 (σ(y1) R5 σ(w1)) = u2 R6 (u2 R5 w1) et
σ(b2) = σ(u2 R6 v2) = σ(u2) R6 σ(v2) = u2 R6 (u2 R5 w1)

Notation:
si le mgu de deux termes t1 et t2 est égal à : $\{(u_1, v_1), \ldots, (u_n, v_n)\}$, on notera $\sigma_{(t1, t2)}$ cette substitution et on écrira par abus de langage
$\sigma_{(t1, t2)} = \{(u_1, v_1), \ldots, (u_n, v_n)\}$,
De même, si l'unification des deux couples de variables (t1, t2) et (t'1, t'2) est égal à : $\{(u_1, v_1), \ldots, (u_n, v_n)\}$, on notera $\sigma_{((t1, t2), (t'1, t'2))}$ cette substitution et on écrira de même: $\sigma_{((t1, t2), (t'1, t'2))} = \{(u_1, v_1), \ldots, (u_n, v_n)\}$,

a.2.3) Propriété fondamentale de l'unificateur principal

On a vu que le mgu de deux termes permet de se prononcer sur leur possibilité d'être instancié par le même objet. En fait, cette opération nous donne également les conditions sur les termes atomiques pour que cette égalité d'instances se produise.

Introduisons d'abord la fonction VAL_ETENDU.
Cette fonction est définie, sur l'ensemble des termes que l'on peut construire avec les constantes et les relations présentes dans r1 ou r2 (appelé l'ensemble $T_\Sigma[X]$ au paragraphe précédent), par:
$VAL\_ETENDU_{(l,l'),(r1,r2)}(x) = VAL_{l,r1}(x)$ si x est composé de termes appartenant tous à r1,
$VAL\_ETENDU_{(l,l'),(r1,r2)}(x) = VAL_{l',r2}(x)$ si x est composé de termes appartenant tous à r2,
$VAL\_ETENDU_{(l,l'),(r1,r2)}(x\ R\ y)$
$= VAL\_ETENDU_{(l,l'),(r1,r2)}(x)\ R\ VAL\_ETENDU_{(l,l'),(r1,r2)}(y)$
si x et y ne sont pas tous les deux composés de termes appartenant tous à la même règle.

Maintenant, on peut énoncer la propriété fondamentale de l'unification de 2 termes:

Propriété fondamentale de l'unificateur principal

Soit t1 et t2 deux termes présents respectivement dans les règles r1 et r2.

Soit σ le mgu de (t1,t2), si σ est égal à (défini par): $\{(u_1, v_1), \ldots, (u_n, v_n)\}$ alors:

et si $\exists$ l,l' tq $VAL_{l,r1}(t1) = VAL_{l',r2}(t2)$,

alors $\forall i$, $VAL_{\alpha,\beta}(u_i) = VAL\_ETENDU_{(l,l'),(r1,r2)}(v_i)$

avec $(\alpha, \beta) = (l, r1)$ si $u_i \in V(t1)$, et $(\alpha, \beta) = (l', r2)$ si $u_i \in V(t2)$

Justification: Par définition, toute instanciation commune à t1 et t2 est une (semi)unification de t1 et t2. Elle se définit donc par composition à partir de σ et d'une autre semi-unification.

Donc, supposons que le mgu de t1 et t2 soit composé de paires indépendantes (u1, v1) , on sait donc que σ(u1) = v1 et σ(v1) = v1 .

Et, par définition, i = $VAL_{l,r1}$ Union $VAL_{l',r2}$ est une substitution telle i(t1) = i(t2), c'est donc est un semi-unificateur de t1 et t2. Donc il s'écrit i = i' σ avec i' une instanciation.

Donc $i(u1) = VAL_{\alpha,\beta}(u1) = i'(\sigma(u1)) = i'(x2)$

et $i(v1) = VAL_{\alpha',\beta'}(v1) = i'(\sigma(v1)) = i'(x2)$

donc si $\sigma(x1) = x2 \Rightarrow VAL_{\alpha,\beta}(u_i) = VAL_{\alpha',\beta'}(v_i)$

Exemple: Reprenons le cas 2 de l'exemple précédent. On a vu que

$\sigma_{(b1, b2)}$ = {(y1, u2), (v2, u2 R5 w1)} avec b1 = y1 R6 (y1 R5 w1) et b2 = u2 R6 v2

Or $VAL_{i,r1}(b1) = VAL_{j,r2}(b2)$, équivaut à écrire:

$VAL_{i,r1}(y1)$ R6 ($VAL_{i,r1}(y1)$ R5 $VAL_{i,r1}(w1)$) = $VAL_{j,r2}(u2)$ R6 $VAL_{j,r2}(v2)$

c'est-à-dire : $-VAL_{i,r1}(y1) = VAL_{j,r2}(u2)$

$- VAL_{i,r1}(y1)$ R5 $VAL_{i,r1}(w1) = VAL_{j,r2}(v2)$

Et, en prenant le mgu, on a directement:

- $VAL_{i,r1}(y1) = VAL\_ETENDU_{(i,j),(r1,r2)}(u2) = VAL_{j,r2}(u2)$
- $VAL_{j,r2}(v2) = VAL\_ETENDU_{(i,j),(r1,r2)}(u2$ R5 $w1) = VAL_{i,r1}(u2)$ R5 $VAL_{i,r1}(w1)$

ce qui est bien la même chose.

a.2.4) Définition d'un conflit

Nous allons maintenant pouvoir passer à la notion de conflit. On suppose pour simplifier l'écriture que le type physique du prédicat est monovalué. S'il ne l'est pas, autant de conflits sont possibles qu'il a de valeurs dans son champ TypePhysique. Ces conflits peuvent être examinés de façon séparée.

Renommage des variables d'une règle

On dira que **la règle r1 est renommé en dehors de la règle r2** si on a réécrit r1 en ayant appliqué à toutes les variables de r1 une substitution, notée $\mathbf{rn_{12}}$, de telle sorte que plus aucune variable de r1 n'ait un nom identique à une variable de r2. On notera $\mathbf{rn_{12}}$(r1) la règle ainsi réécrite.

Définition d'un conflit, mgu du conflit

Soit r1 ayant comme conséquent :
(c1) : x1 R y1 , avec x1 et y1 des termes
et r2 tel que $\mathbf{rn_{21}}$**(r2)** a comme conséquent :
(c2) : x2 R y2 , avec x2 et y2 des termes ;
et R tel que TypePhysique(R) ∈ {fonctionnel, injectif, antisymétrique}.
On dira que **r1 est en conflit avec r2 sur leur conséquent c1 et c2 à propos de TypePhysique(R)** dans les trois cas suivants:

- si Type Physique(R) = **fonctionnel,**
  il existe un mgu σ de x1 et x2, et σ(y1) ≠ σ(y2)
- si Type Physique(R) = **injectif,**
  il existe un mgu σ de y1 et y2, et σ(x1) ≠ σ(x2)
- si Type Physique(R) = **antisymétrique**
  il existe un mgu σ de (x1,x2) et (y1,y2)

σ est appelé un mgu du conflit.
On notera **C(r1, r2, c1, c2)** ce conflit
et on dira que l'objet de ce conflit est R, ie **OBJET(C(r1, r2, c1, c2) = R**

Remarque: Dans le cas ou le prédicat est de suffisance contextuelle:

1°) L'unification doit alors avoir lieu en rajoutant les contextes. Par exemple, si R est fonctionnelle et de suffisance contextuelle, il faut pour qu'il y est conflit que les contextes soient identiques, ie par exemple, qu'on ait deux conséquents de la forme: (c1): (x1 R y1) G c1 dans r1, et (c2): (x2 R y2) G c2 dans $\mathrm{rn_{21}}$(r2)
on que le mgu de l'ensemble ((x1, x2), (c1, c2)) existe

2°) Les prédicats complémentaires (par exemple, ici, G est complémentaire de R - voir 3)c)c.2)) ne doivent pas apparaître dans des conséquents de règle sans que soit également explicité le (ou un des) prédicat(s) dont il est complémentaire[18].

---

[18] Et cela pour la raison suivante. Supposons que G soit un prédicat complémentaire d'un prédicat R de type physique fonctionnel. On sait qu'on ne peut avoir les deux expressions (a R b) G c et (a R d) G c
Supposons qu'il soit possible d'écrire la règle r1: SI (X) Q (Y) ALORS (X) G (Y)
(suite page suivante)

Exemples de conflits :
(les symboles en majuscule sont des constantes, ceux en minuscule sont des variables. On n'omettra parfois les parenthèses dans la notation des variables pour soulager l'écriture)

1°) Les règles ayant les conséquents suivants sont en conflit

(c1) : F(x) = A (c2) : F(z) = B
avec F un prédicat de type physique fonctionnel
En effet σ = {(x, z} et σ(A) ≠ σ(B) puisque σ(A) = A et σ(B) = B[19]

(c1) : ((x1 R1 A) F B) R2 (x2 R3 A) (c2) : ((z2 R1 y2) F C) R2 (z2 R3 y2)
avec F un prédicat de type physique contextuellement fonctionnel, et R3 son contexte.
En effet σ = {(x1, z2), (y2, A),(x2, z2)}
on a bien σ(x1 R1 A) = σ(z2 R1 y2), σ(x2 R3 A) = σ(z2 R3 y2) et σ(B) ≠ σ(C)

(c1) : (x1 R1 A) R2 B (c2) : x2 R2 (A R1 y2)
avec R2 un prédicat de type physique antisymétrique
En effet σ = {(x1, A), (y2, A), (x2, B)}
on a bien σ(x1 R1 A) = σ(x2) et σ(B) = σ(A R1 y2)

2°) Les règles ayant les conséquents suivants ne sont pas en conflit

(c1) : ((z1 R1 A) R2 y1) F y1 (c2) : (z2 R2 A) F A
avec F un prédicat de type physique fonctionnel
En effet σ = {(z2, z1 R1 A), (y1, A)}
et σ(y1) = A = σ(A)

(c1) : ((x1 R1 A) R2 y1) R3 (z1 R4 y1) (c2) : (x2 R4 B) R3 (x2 R2 A)
avec F un prédicat de type physique antisymétrique
En effet, on cherche le mgu de (((x1 R1 A) R2 y1, x2 R2 A), (z1 R4 y1, x2 R4 B))
On déduit les couples (x2, x1 R1 A) , (z1, x1 R1 A) , (y1, A)
puis on peut unifier σ_i(y1) = A avec B, d'où échec.

(c1) : R1(x1) = A (c2) : R1(x2) = (C R2 (y2))
avec R1 un prédicat de type physique injectif
En effet, l'unification entre A et (C R2 (y2)) est impossible.

Comment savoir si cette règle ne permet pas de rajouter des instances de R contradictoires avec celles déjà existantes? Il faudrait examiner quels sont les valeurs d'instanciation possibles de (X), en regardant les règles déduisant la relation Q.
Ce travail nous a paru trop compliqué en regard de l'avantage qu'il fournit.

[19] N'oublions pas que l'on note de la même façon la même façon une substitution s et son prolongement $\underline{s}$ à l'ensemble des termes, et que $\underline{s}$(c) = c si c ∈ Σ

a.2.5) Contradiction et Conflit

Lemme 1:
Si un couple de règles peut générer des faits contradictoires par coexistence, alors il existe un conflit entre ces règles.

Démonstration
(on ne fait la démonstration que dans le cas ou le conflit est du à un type physique fonctionnel. Les deux autres cas se déduisent trivialement de celui-ci)
Si r1 et r2 génère des faits contradictoires par coexistence, alors, d'après la CN de contradiction:

- r1 contient le conséquent (c1) : x1 R y1,
- r2 contient le conséquent (c2) : x2 R y2,
- R est de type physique fonctionnel,
- il existe deux inférences INF1 et INF2 tels que il existe i et j tels que

$VAL_{i,r1}(x1) = VAL_{j,r2}(x2)$ et $VAL_{i,r1}(y1) <> VAL_{j,r2}(y2)$

Or, a) si $VAL_{i,r1}(x1) = VAL_{j,r2}(x2)$, on sait que le mgu σ de x1 et x2 existe et par définition $\sigma(x1) = \sigma(x2)$

b) et si on avait $\sigma(x1) = \sigma(x2) \Rightarrow \sigma(y1) = \sigma(y2)$ ie si $(y1,y2) \in \sigma$

alors $VAL_{i,r1}(x1) = VAL_{j,r2}(x2) \Rightarrow VAL_{i,r1}(y1) = VAL_{j,r2}(y2)$
(propriété fondamentale du mgu)
or $VAL_{i,r1}(y1) <> VAL_{j,r2}(y2)$ donc $(y1,y2) \notin \sigma$
=> soit $\sigma(y1) <> \sigma(y2)$, soit y1 ou y2 n'appartiennent pas tous les deux à $D(\sigma)$ et dans ce cas on peut fixer arbitrairement la valeur de $\sigma(y1)$ ou $\sigma(y2)$ et avoir ainsi $\sigma(y1) <> \sigma(y2)$

a.3) Règle résolvante d'un conflit

On a donc un moyen de repérer les contradictions potentielles. Le problème est maintenant de savoir si les couples de règles en conflit sont vraiment contradictoires. Ce sera la fonction de la règle résolvante.

Notation: On note $\mathbf{rn_\sigma(r)}$ la règle réécrite en remplaçant toutes les variables de r par leur image par σ.

Construction de la Règle Résolvante
On a vu que pour qu'il y ait génération de deux faits contradictoires par coexistence, il fallait que les deux règles puisse se déclencher sur la même base de faits et que les deux termes en place objet des conséquents en conflit puissent prendre la même valeur. Pour tester ces deux conditions, **on réunit en une seule règle les deux règles en conflit en servant des relations entre variables établies par le mgu du conflit.**

Soit donc C(r1, r2, c1, c2) un conflit et σ le mgu du conflit.
On définit **la règle résolvante r1.c1,2.c2** par :
$PREM(r1.c1,2.c2) = PREM(rn_\sigma(r1))$ ET $PREM(rn_\sigma(rn_{21}(r2)))$
$CONS(r1.c1,2.c2) = CONS(rn_\sigma(r1))$ ET $CONS(rn_\sigma(rn_{21}(r2)))$

Exemple:

REGLE r1	REGLE r2
SI	SI
B F1 (y)	(x) F1 A
(x) R4 (y)	(x) R2 (y): (z)
(x) R5 D	(y) R3 C
ALORS	ALORS
B R6 (y) : (z)	(x) R6 (z) : (t)
(z) F (y)	(t) F E

et F de type physique fonctionnel

On choisit par exemple rn_{21} = {(x, x2), (y, y2), (z, z2), (t, t2)}

On a $\sigma_{(z,t2)}$ = {(x2, B), (y, B R2 y2)}

---> règle résolvante: regle r1.1,2.1

```
SI
    B F1 (B R2 (y2))        ; noter la nouvelle forme de y dans r1;
    (x) R4 (B R2 (y2))
    (x) R5 D
    B F1 A                  ; identification de (x) de r2 avec B;
    B R2 (y2) : (z2)        ; (z2) = nouvelle forme du (y) de r1;
    (y2) R3 C
ALORS
    B R6 (B R2 (y2)) : (t2)
    (t2) F (B R2 (y2))
    B R6 (z2) : (t2)        ;c'est-à-dire B R6 (B R2 (y2)) : (t2) à
nouveau;
    (t2) F E
```

Notation: Par abus de langage, on note σ(c1) et σ(c2) les deux conséquents de r1.c1,2.c2 correspondant aux conséquents c1 et c2 de r1 et r2.

Propriété fondamentale de la règle résolvante

Soit C(r1, r2, c1, c2) un conflit,
(FA(i, r1) et FA(j, r2) sont les faits correspondants au déclenchement de r1 et r2 ayant entraîné grâce aux conséquents c1 et c2 la déduction d'une paire de faits contradictoires par coexistence)
<==>
(FA(i, r1) UNION FA(j, r2) est un ensemble (pas forcément non contradictoire) de faits qui satisfait la règle résolvante r1.c1,2.c2 et lui permet de déduire dans un même déclenchement deux faits contradictoires par coexistence grâce aux conséquents σ(c1) et σ(c2))

Démonstration

On note $\{v_{1,1}, \ldots, v_{1,n}\}$ les variables de r1 et $\{v_{2,1}, \ldots, v_{2,p}\}$ les variables de rn_{21}(r2).

On sait que ces variables seront transformées en $\sigma(v_{1,i})$ et $\sigma(v_{2,i})$ dans r1.c1,2.c2.

Or, on sait (propriété fondamentale du mgu) que

si $(u_i, v_i) \in \sigma_{(x1,x2)}$, $VAL_{\alpha,\beta}(u_i) = VAL\_ETENDU_{((i,j)(r1,r2))}(v_i)$
avec (α, β) = (i, r1) ou (j, r2) suivant la nature de u_i

Donc, si on remplace u_i par $\sigma(u_i)$ dans la règle auquel il appartient, son instanciation par $VAL_{\alpha,\beta}(u_i)$ est une instanciation de $\sigma(u_i)$.
De plus, les deux règles ne sont pas logiquement exclusives puisqu'elles ont pu produire une paire de faits contradictoires par coexistence.
On construit donc à partir de FA(i, r1) UNION FA(i, r2) l'instanciation globale de r1.c1,2.c2 suivante :
IG(k, r1.c1,2.c2) =

$\{(\sigma(v_{1,1}), VAL_{i,r1}(v_{1,1})), (\sigma(v_{1,2}), VAL_{i,r1}(v_{1,2})), \ldots, (\sigma(v_{1,n}), VAL_{i,r1}(v_{1,n})),$
$(\sigma(v_{2,1}), VAL_{j,r2}(v_{2,1})), (\sigma(v_{2,2}), VAL_{j,r2}(v_{2,2})), \ldots, (\sigma(v_{2,p}), VAL_{j,r2}(v_{2,p}))\}$

Cette instanciation est possible dans FA(i, r1) UNION FA(i, r2) puisque cet ensemble contient les faits correspondant aux valeurs utilisés dans cette instanciation.

A cette instanciation globale correspond le déclenchement numéro k (on ne connait pas la valeur de k mais on peut supposer qu'il est égal à 1 par exemple) dans lequel, si, par exemple, les deux conséquents de r1 et r2 générant les faits contradictoires par coexistence étaient (après renommage de r2):
(c1) : x1 F y1 et (c2) : x2 F y2 , avec F un prédicat de type physique FONCTIONNEL:
on a:

$VAL_{k,r1.c1,2.c2}(\sigma(y1)) = VAL_{i,r1}(y1)$

$VAL_{k,r1.c1,2.c2}(\sigma(y2)) = VAL_{j,r2}(y2)$ et on sait que $VAL_{i,r1}(y1) <> VAL_{j,r2}(y2)$

==> $\sigma(y1)$ s'instancie par une valeur distincte de $\sigma(y2)$,
et

- $VAL_{k,r1.c1,2.c2}(\sigma(x1)) = VAL_{i,r1}(x1)$
- $VAL_{k,r1.c1,2.c2}(\sigma(x2)) = VAL_{i,r1}(x2)$

donc $VAL_{k,r1.c1,2.c2}(\sigma(x1)) = VAL_{k,r1.c1,2.c2}(\sigma(x1))$
ie $\sigma(x1)$ s'instancie par la même valeur que $\sigma(x2)$,

==> lors de ce déclenchement, les deux conséquents $\sigma(c1) = \sigma(x1)$ F $\sigma(y1)$ et
$\sigma(c2) = \sigma(x2)$ F $\sigma(y2)$ de r1.c1,2.c2 déduiront deux faits contradictoires par coexistence.

$\Leftarrow$: La réciproque est évidente par construction de r1.c1,2.c2 : tout ensemble d'objets associés à un déclenchement ayant déduit des faits contradictoires par coexistence contient de quoi déclencher r1 et r2 en leur faisant déduire des faits contradictoires par coexistence.

Exemple

REGLE	r1	REGLE	r2
SI	R (x) = (y) : (z)	SI	P (x) = (y)
ALORS	F A = (z)	ALORS	F B = (x)

et F de type physique INJECTIF

- On prend comme ensemble de faits ayant permis des déductions de faits contradictoires par coexistence:
FA(1, r1) = {A R C} et FA(1, r2) = {(A R C) P D}

correspondant aux instanciations globales:

IG(1, r1) dans FA(1, r1)

x	A
y	C
z	A R C

déduisant: A F (A R C)

IG(1, r2) dans FA(1, r2)

x	A R C
y	D

déduisant : B F (A R C)

- on sait que le mgu du conflit est : $\sigma_{(z, x2)}$ = {(x2, x R y)}

- donc la règle résolvante est:

REGLE r1.1,2.1
SI R (x) = (y) : (x2) ; remplacement de z par x2 dans r1;
P ((x) R (y)) = (y2) ; la nouvelle forme de x2 ;
ALORS F A = (x2)
F B = ((x) R (y))

- l'instanciation globale de la règle résolvante est:

IG(r1.1,2.1) dans {A R C, (A R C) P D}

x	A
y	C
x2	A R C
y2	D

déduisant: A F (A R C)
B F (A R C)

a.4) Résolution de conflit

a.4.1) Résolution de conflit par indéclenchabilité de la règle résolvante

Indéclenchabilité d'une règle sous réserve de E_MR

Une règle r est indéclenchable si et seulement si il n'existe pas d'ensemble de faits **non contradictoire** qui la satisfasse.

Autrement dit, **la règle contient un ensemble de prémisses exclusives, c'est-à-dire un ensemble de prémisse:**

- **logiquement exclusives** , par exemple (X) > 1 et (X) < 0,

OU - sémantiquement exclusives, ie dont les propriétés des prédicats qui les composent les rendent exclusives.

Si cet ensemble est exclusif à cause d'un ensemble de métarelations E_MR, on dira que la règle est indéclenchable sous réserve de E_MR.

Autrement dit, si la règle se déclenche, **c'est que l'une des métarelations de E_MR est violée par la BF courante.**

Exemple: Soit la règle

```
REGLE r
SI   A F1 (x)      ; prémisse (p1) ;
     A F1 (y)      ; prémisse (p2) ;
     (x) F2 B      ; prémisse (p3) ;
     (y) F2 C      ; prémisse (p4) ;
ALORS ...
```

avec F1 et F2 de type physique fonctionnel

Des prémisses (p1) et (p2), on déduit l'égalité : $x =_S y$ (voir le 3)b) pour la définition de $=_S$ et $<>_S$) car x et y sont les images par une même fonction d'un même terme.

Des prémisses (p3) et (p4), on déduit la relation : $x <>_S y$ (x et y sont les antécédents par une même fonction de deux termes différents).

Donc la règle n'est satisfiable par aucun ensemble de faits non contradictoire, elle contient l'ensemble {p1, p2, p3, p4} de prémisses sémantiquement exclusives, donc est indéclenchable sous réserve de E_MR = {F1 TypePhysique FONCTIONNEL, F2 TypePhysique FONCTIONNEL}.

MISE EN OEUVRE

Un ensemble de prémisses exclusives peut être détectée en quatre étapes:

1°) On explicite les propriétés des concepts

Cette phase consiste à construire les ensembles de valeurs d'instanciations possibles des variables représentant un concept défini en extension.

Pour toute prémisse (p) : x R y

A) si DomaineConceptuelObjet (R) = (co) et (co) DefiniPar (Liste)
alors DomaineDInstanciationPossible(x) = (Liste)

B) si DomaineConceptuelValeur(R) = (cv) et (cv) DefiniPar (Liste)
alors DomaineDInstanciationPossible(y) = (Liste)

Une règle possédant une variable ayant son domaine d'instanciation connu sera dupliquée en autant de règles que ce domaine possède de valeurs, et dans chaque nouvelle règle, la variable en question est remplacée par une valeur différente de son ensemble de valeurs d'instanciations possibles.

2°) On explicite les propriétés des prédicats
et on rajoute à la règle les prémisses correspondantes
(on ne cite ici que les propriétés principales utilisables)

A) si la règle contient les prémisses (p1) : x1 R y1 et (p2) : x2 R y2
1) si TypePhysique(R) = FONCTIONNEL
si x1 $=_S$ x2 alors on rajoute la prémisse : y1 = y2 (et on a y1 $=_S$ y2)
si y1 $<>_S$ y2 alors on rajoute à la prémisse : x1 <> x2 (et on a x1 $<>_S$ x2)
2) si TypePhysique(R) = INJECTIF
si y1 $=_S$ y2 alors on rajoute la prémisse : x1 = x2 (et on a x1 $=_S$ x2)
si x1 $<>_S$ x2 alors on rajoute la prémisse : y1 <> y2 (et on a y1 $<>_S$ y2)
3) si TypePhysique(R) = ANTISYMETRIE
si x1 $=_S$ y2 alors on rajoute la prémisse : y1 <> x2 (et on a y1 $<>_S$ x2)
si y1 $=_S$ x2 alors on rajoute la prémisse : x1 <> y2 (et on a x1 $<>_S$ y2)

B) si la règle contient les prémisses (p1) : x1 R y1 et (p2) : x2 R y2 ' INEXISTANT
1) si TypePhysique(R) = SYMETRIE
si x1 $=_S$ y2 alors on rajoute la prémisse : y1 <> x2 (et on a y1 $<>_S$ x2)
si y1 $=_S$ x2 alors on rajoute la prémisse : x1 <> y2 (et on a x1 $<>_S$ y2)
2) si TypePhysique(R) = REFLEXIF (à droite)
si y1 $=_S$ x2 alors on rajoute la prémisse : x2 <> y2 (et on a x2 $<>_S$ y2)
si x2 $=_S$ y2 alors on rajoute la prémisse : y1 <> x2 (et on a y1 $<>_S$ x2)

C) si la règle contient les prémisses (p1) : x1 P y1 et (p2) : x2 Q y2
1) si P et Q DIFFERENTS(-objet)
si x1 $=_S$ x2 alors on rajoute la prémisse : y1 <> y2 (et on a y1 $<>_S$ y2)
si y1 $=_S$ y2 alors on rajoute la prémisse : x1 <> x2 (et on a x1 $<>_S$ x2)
2) si P et Q INCOMPATIBLES(-objet) ou si DCO[20](P) $\cap$ DCO(Q) = $\varnothing$
alors on rajoute la prémisse : x1 <> x2 (et on a x1 $<>_S$ x2)

3°) On fait les fermetures transitives sur les comparateurs et sur la propriété $=_S$:
Par exemple, si la règle contient les prémisses
(p1) : x1 > y1 et (p2) : y1 > z1 , on rajoute la prémisse (p3) : x1 > z1
ce genre de réécriture comprenant également la prise en compte de l'ordre sur les nombres (on déduit x > y de x > 1 et y < 0 par exemple)
fermeture de $=_S$: Si l'on a déduit x1 $=_S$ x2 et x2 $=_S$ x3, on déduit x1 $=_S$ x3

4°) On cherche une prémisse aberrante
par exemple 3 <> 3, ou (X) > (X) etc ..
Si la ou les prémisses aberrantes ont été obtenues en se servant de l'ensemble de métarelations {R_1 PI_{k1} v_{l1}, ... ,R_n PI_{kn} v_{kn}, P_1 MR_{l1} Q_1, ... , P_k MR_{lk} Q_k} dans lequel :

---

[20] abréviation de DomaineConceptuelObjet

R_i PI_{ki} v_{ki} signifie que la valeur du champ PI_{ki} pour le prédicat R_i est v_{ki} (propriété intrinsèque), et P_j MR_{ijk} Q_j signifie qu'on a la métarelation MR_{ijk} entre P_j et Q_j (relation entre prédicats),
alors on dira que **la règle est indéclenchable sous réserve de**
E_MR = {R_1 PI_{k1} v_{l1}, ... ,R_n PI_{kn} v_{kn}, P_1 MR_{l1} Q_1, ... , P_k MR_{lk} Q_k}

Exemple:
Dans l'exemple précédent, on aurait d'abord rajouté la prémisse
(p5) : x = y grâce au A)1), puis la prémisse (p6) : x <> y aussi grâce au A)1), et ensuite détecté que ses deux prémisses étaient aberrantes (d'après le 3°)).

Application à la résolution de conflit:
Résolution de conflit par indéclenchabilité de la règle résolvante
Soit C(r1, r2, c1, c2) un conflit entre r1 et r2, et r1.c1,2.c2 la règle résolvante de ce conflit.
Si r1.c1,2.c2 est indéclenchable (sous réserve d'un ensemble E_MR de métarelations), on dira que C(r1, r2, c1, c2) est résolu par indéclenchabilité de la règle résolvante (sous réserve de E_MR).

Lemme 2
Soit C(r1, r2, c1, c2) est résolu par indéclenchabilité de la règle résolvante (sous réserve d'un ensemble E_MR de métarelations)
alors r1 et r2 ne peuvent être satisfaites dans un même ensemble non-contradictoire de faits et déduire grâce aux conséquents c1 et c2 des faits contradictoires par coexistence.

Démonstration
si r1.c1,2.c2 est indéclenchable, elle est insatisfiable, donc il n'existe pas d'ensemble non-contradictoire de faits qui la satisfasse. Donc les seuls ensembles de faits FA(i, r1.c1,2.c2) qui la satisfont sont contradictoires.
Or on sait que si deux ensembles de faits FA(i, r1) et FA(j, r2) ont permis de déclencher respectivement r1 et r2, entraînant la déduction d'une paire de faits contradictoires grâce aux conséquents c1 et c2, alors leur réunion satisfait r1.c1,2.c2 (propriété fondamentale de la règle résolvante)
donc la réunion d'ensemble du type FA(i, r1) et FA(j, r2) est forcément contradictoire.
Or si l'on veut que r1 et r2 puissent se déclencher sur un même ensemble de faits, et déduisent alors des faits contradictoires par coexistence grâce aux conséquents c1 et c2, alors cet ensemble contient au moins FA(i, r1) et FA(j, r2), donc est contradictoire.

Exemple; Soient les deux règles:

REGLE r1
SI F1(x) = a
ALORS F2(x) = a

REGLE r2
SI F1(x) = b
ALORS F2(x) = b

avec F1 et F2 de type physique FONCTIONNEL
-> On construit alors : REGLE r1.1,2.1
SI F1(x) = a
F1(x) = b
ALORS F2(x) = a
F2(x) = b

-> La règle résolvante est indéclenchable : en effet, on génère la prémisse : X <> X lors de l'application de la règle de réécriture concernant la monovaluation des fonctions (cf 3°)).
-> En effet, les deux règles ne peuvent générer de faits contradictoires par coexistence qu'a partir de faits contradictoires par coexistence. Le couple de règles n'est donc pas contradictoire.

a.4.2) Résolution de conflit par égalité d'instances

même instanciation entre deux termes d'une règle sous réserve de {P1, ..., Pn}
Si on a déduit, lors de l'application des règles présentées au paragraphe précédent que deux termes y1 et y2 de la règle sont tels que y1 $=_s$ y2, on dira que ces deux termes ont même instanciation sous réserve de {...,R_i PI_{ki} v_{ki}, ... , P_j MR_{lj} Q_j, ...}
On sait alors que si l'ensemble des propriétés est respecté, alors
$\forall i \quad VAL_{i,r1}(y1) = VAL_{i,r1}(y2)$

Exemple: Soit la partie-prémisse de règle

```
SI  F1 A = (x)     ; prémisse (p1);
    F1 A = B       ; prémisse (p2);
    F2 A = (y1)    ; prémisse (p3);
    F2(x) = (y2)   ; prémisse (p4);
    ...
```

avec F1 et F2 des prédicats de type physique fonctionnel
Des prémisses (p1) et (p2), on déduit la relation : (x) = B
De la relation ci-dessus de (p3) et de (p4), on déduit la relation: (y1) = (y2)
Donc dans toute instanciation globale de r, (y1) est instancié avec la même valeur que (y2), on dit que (y1) a la même instanciation que (y2) sous réserve de {Type Physique(F1) = FONCTIONEL, Type Physique(F2) = FONCTIONEL}

Application à la résolution de conflit:

On a vu que dans le cas de conflit à propos d'un type physique fonctionnel ou injectif, on a deux conditions nécessaires pour qu'il y ait contradiction. Il faut d'une part que le mgu σ du conflit existe, et d'autre part que les valeurs par σ de deux termes soit différentes. On a des conditions du style: il existe un mgu σ de x1 et x2,

et $\sigma(y1) \neq \sigma(y2)$. Ce qui revient à exiger les valeurs d'instanciations de $\sigma(y1)$ et $\sigma(y2)$ dans la règle résolvante soient différentes.

Or, si on arrive à montrer que dans la règle résolvante $\sigma(y1) =_s \sigma(y2)$, alors on sait que $\forall i \quad VAL_{i,r1}(\sigma(y1)) = VAL_{i,r1}(\sigma(y2))$

donc les valeurs d'instanciations de $\sigma(y1)$ et $\sigma(y2)$ dans la règle résolvante ne sont jamais différentes, donc la contradiction qui ait à l'origine de cette règle résolvante n'est pas effective. Si les deux règles n'avait que ce conflit entr'elles, elles ne génèreront pas de faits contradictoires par coexistence.

Résolution de conflit par égalité d'instances

Soit C(r1, r2, c1, c2) un conflit entre r1 et r2 avec (après renommage de r2)

(c1) : x1 R1 y1 et (c2) : x2 R1 y2, et σ le mgu du conflit

On dira que le conflit est résolu par égalité d'instances

(sous réserve de $\{..., R_i\ PI_{ki}\ v_{ki'} \ldots, P_j\ MR_{lj}\ Q_{j'} \ldots\}$) si:

- R1 de type physique fonctionnel et $\sigma(y1) =_s \sigma(y2)$
- R1 de type physique injectif et $\sigma(x1) =_s \sigma(x2)$

On a alors $\sigma(c1) =_s \sigma(c2)$. En effet, on sait que

$\sigma(c1)$ désigne le conséquent: $\sigma(x1)$ F $\sigma(y1)$

$\sigma(c2)$ désigne le conséquent: $\sigma(x2)$ F $\sigma(y2)$, et:

- dans le premier cas $\sigma(x1) = \sigma(x2)$ et $\sigma(y1) =_s \sigma(y2)$
- dans le deuxième cas $\sigma(x1) =_s \sigma(x2)$ et $\sigma(y1) = \sigma(y2)$

Donc dans tous les cas et quelque soit i:

$VAL_{i,r1.c1,r2.c2}(\sigma(x1)\ F\ \sigma(y1)) = VAL_{i,r1.c1,r2.c2}(\sigma(x2)\ F\ \sigma(y2))$, donc $\sigma(c1) =_s \sigma(c2)$

Lemme 3

Soit C(r1, r2, c1, c2) un conflit entre r1 et r2 résolu par égalité d'instances
(sous réserve d'un ensemble E_MR de métarelations)
Alors r1 et r2 ne peuvent être satisfaites dans un même ensemble de faits non-contradictoire et déduire grâce aux conséquents c1 et c2 des faits contradictoires par coexistence.

Démonstration

On sait que si C(r1, r2, c1, c2) est résolu par égalité d'instances, alors $\sigma(c1) =_s \sigma(c2)$

or $\sigma(c1) =_s \sigma(c2) \Rightarrow \forall i \quad VAL_{i,r1.c1,2.c2}(c1) = VAL_{i,r1.c1,2.c2}(c2)$

$\Rightarrow$ il n'existe pas d'ensemble non contradictoire de faits tel que la règle résolvante soit satisfaite dans cet ensemble et que le déclenchement correspondant permettent de déduire des faits contradictoires par coexistence grâce aux conséquents $\sigma(c1)$ et $\sigma(c2)$ puisque ceux-ci s'instancient toujours par une valeur identique

$\Rightarrow$ d'après la propriété fondamentale de la règle résolvante, il n'existe pas d'ensemble non contradictoire de faits qui satisfasse r1 et r2 et leur permette de déduire des faits contradictoires par coexistence grâce aux conséquents c1 et c2.

a.5) Condition Suffisante de non contradiction dans les règles

Lemme 4
Si tous les conflit entre r1 et r2 sont:
- soit résolu par indéclenchabilité de la règle résolvante
- soit résolu par égalité d'instances
Alors r1 et r2 sont non contradictoires

Pour chaque conflit, si {..., $PI_i(Rj)$, ... , $MR_k(Pm,Qm)$, ...} est l'ensemble des propriétés nécessaires à la preuve de la résolution de ce conflit, on dira que **ce conflit est résolu sous réserve de {..., $PI_i(Rj)$, ... , $MR_k(Pm, Qm)$, ...}**.

Démonstration
Par l'absurde à partir des lemmes 1, 2 et 3.
Supposons que r1 et r2 forment un couple contradictoire, cela veut dire qu'il existe c1 et c2 deux conséquents de r1 et r2 et EF un ensemble non contradictoire de faits tel que:
- r1 est satisfiable dans EF
- r2 est satisfiable dans EF
- que les déclenchements correspondant permettent de déduire grâce aux conséquents c1 et c2 deux faits contradictoires par coexistence.

On sait qu'alors il existe un conflit C(r1, r2, c1, c2) entre r1 et r2 (lemme 1)
et que si r1 et r2 sont satisfaites dans un même ensemble non-contradictoire de faits et déduisent grâce aux conséquents c1 et c2 des faits contradictoires par coexistence, alors la règle résolvante r1.c1,2.c2 du conflit correspondant, qui possède les deux conséquents $\sigma(c1)$ et $\sigma(c2)$ (avec σ le mgu du conflit), est :
- non indéclenchable (contraposée du lemme 2)
- telle que $\neg(\sigma(c1) =_S \sigma(c2))$ (contraposée du lemme 3)

Or, par hypothèse, aucune règle résolvante de conflit entre r1 et r2 ne respecte ces deux conditions.
=> impossible.

Remarque: Cette condition n'est pas nécessaire. Par exemple, les deux règles

REGLE r1		REGLE r2	
SI	F1(X) = (Y)	SI	F3(X) = (Y)
ALORS	F2(X) <- (Y)	ALORS	F2(X) <- (Y)
	F1(X) = a		F3(X) = a

avec F1, F2 et F3 des prédicats de type physique fonctionnel.
Ces deux règles, apparemment contradictoires, ne le sont en fait pas à cause des autorécursions qui font qu'au cours de l'inférence, les triplets contradictoires de la forme x F2 y sont réécrits x F2 a dans les deux règles.

a.6) Critère de conformité d'un prédicat à son type physique:
cas FONCTIONNEL, INJECTIF et ANTISYMETRIQUE

Nous allons maintenant conclure. On avait au début de cette partie remarqué que deux types de contradictions dans les règles pouvait rendre un prédicat non conforme à son type physique, pour les valeurs de ce type envisagées ici:

- les contradictions intrinsèques de type 2:

ie les règles de la forme par exemple: REGLE r1 SI F(X) = a ALORS F(X) = b
avec F un prédicat de type physique fonctionnel.

- les couples contradictoires de règles.

Nous ne nous sommes intéressé qu'au deuxième type de contradiction pour la raison suivante: Par définition de la contradiction intrinsèque, le prédicat objet de la contradiction (ici F) est déductible, donc ne peut apparaître en BFi (en effet, une BFi valide ne contient pas d'instance d'une relation déductible, cf 3)c)c.4)).
Donc, pour que ce type de règle puisse générer, à partir d'une BFi valide, un fait contradictoire avec le contenu de la BF courante, il faut qu'il existe d'autres règles (notons-les r2i) ayant déduit le fait qui va être contredit (dans notre exemple, un objet du type x F a). Donc que les couples de règles (r1, r2i) soient contradictoires.

Et, dans un processus d'inférence à partir d'une BFi valide:

- soit une règle r2i se déclenchera. Puisque les conflits sont résolus, on sait que r1 ne se déclenchera pas,
- soit aucune règle r2i se déclenchera, auquel cas la prémisse de r1 ne sera pas satisfaite, donc la règle ne se déclenchera pas.

Donc, dans aucun cas, la règle ne pourra se déclencher. Elle ne pourra donc en aucun cas générer de faits contradictoires. Le problème de la contradiction intrinsèque ne se pose donc pas vraiment pour les propriétés envisagées ici, .

Nous pouvons maintenant énoncer le critère:

Critère de conformité d'un prédicat à son Type Physique (cas des valeurs: FONCTIONNEL, INJECTIF ou ANTISYMETRIQUE) Un prédicat est conforme à son type physique s'il n'est pas l'objet de conflit ou si tous les conflits dont il est l'objet sont résolus sous réserve d'un ensemble de propriétés opérantes

Justification
Pour que lors d'une inférence à partir d'une BFi valides, l'une de ses propriétés soit violée, il faut que deux règles, notons les r1 et r2 se soient déclenchées et aient déduit deux faits contradictoires par coexistence (rappelons que puisque la BFi ne contient pas d'instances de relations déductibles, la contradiction aura forcément lieu entre deux faits déduits, et non entre un fait déduit et un fait présent en BFi).
On sait qu'il existe alors un conflit entre r1 et r2 correspondant à la déduction de ces deux faits contradictoires par coexistence.
Or, par hypothèse, ce conflit est résolu, sous réserve d'un ensemble de métarelations, noté E_MR1.

Donc, supposons que les deux faits contradictoires par coexistence aient été inférés respectivement aux déclenchements numèro i et j de l'inférence par (respectivement) r1 et r2,
alors on sait que IG(i,r1) $\cup$ IG(j,r2) est un ensemble contradictoire de faits (lemme 2 et 3).
Plus précisément, puisque le conflit est résolu sous réserve de E_MR1, on sait que c'est l'une des propriétés de E_MR1 qui a été violée pour que la contradiction ait eu lieu.
Or ces propriétés sont opérantes, ce qui veut dire qu'on sait qu'elles ne peuvent être violées lors d'une inférence à partir d'une base de faits initiale
$\Rightarrow$ contradiction

On voit donc s'amorcer le principe de la vérification: partant de propriétés "sûres", on prouve la sûreté d'autres propriétés, qui elles-mêmes serviront à prouver sûres de nouvelles propriétés etc ...

a.7) Discussion critique

Cette vérification se fait donc en supposant qu'une base de faits initiale est valide si:

- elle est composé d'un ensemble non-contradictoire de faits
- elle ne contient aucune relation mentionnant un prédicat déductible par les règles que l'on va vérifier. Autrement dit, elle ne comporte que des relations construite avec des prédicats appartenant à l'ensemble Pred_initiaux défini au chapitre 4. Cette deuxième hypothèse est forte. Elle devrait être remplacée par la spécification des BFi valides, en terme par exemple de prédicats et de domaines conceptuels.

De plus, les vérifications sont menées en supposant (et en imposant des critères pour maintenir) une monotonie totale sur l'ensemble des prédicats concernés. Ce choix n'est pas réaliste. Mais tenir compte des possibilités de variation de la présence des relations nécessite obligatoirement de connaître l'allure globale du déroulement de l'inférence. En effet:

- d'une part, il est impératif de savoir, comme avec un ATMS, les conséquences de la destruction d'un fait sur les propriétés prouvées grâce à ce fait, et cela dépend bien sûr du moment auquel est détruit ce fait (par exemple, si a R b empêche la règle déduisant a F b de se déclencher, et que l'on découvre par ailleurs qu'il va être détruit, il faut absolument savoir si au moment de sa destruction, la règle qu'il bloquait est encore activable),
- d'autre part, retrouver sans aucune information supplémentaire, les récursions réelles entre règles d'une BdR est à lui seul un travail de titan (comme l'a prouvé [DORMOY90] et [DORMOY91]), qui ne laisse aucune "place machine" pour d'autres opérations. Il faut donc disposer d'un "plan d'expertise" pour commencer à s'y retrouver, comme l'a d'ailleurs réalisé, dans des travaux plus récent, [BARRIERE90]. Il est hors de doute que c'est la bonne voie. Je ne l'ai pas emprunté par manque de temps et de connaissances sur la représentation interne du moteur qui exploitait les BdR que j'étudiais.

b) Symétrie, Réflexivité à droite et à gauche

b.1) Présentation

Les vérifications qui vont être effectuées maintenant seront beaucoup plus simples. Paradoxalement parce que le problème à résoudre est beaucoup plus compliqué. En effet, il s'agit ici de s'intéresser à la non-contradiction d'un ensemble de règles, correspondant à la création d'une contradiction par absence dans un ensemble de faits. Il faut donc de démontrer la "déclenchabilité" systématique d'une règle dans un contexte donné. Par exemple, si R est symétrique, il faut s'assurer que si l'on déduit le triplet a R b, le triplet b R a est aussi déduit.

Le premier problème est de savoir quand vérifier que R est déduit. Jusqu'à présent, nous n'avons pas tenu compte de la stratégie d'exploitation des règles du moteur. Pour pouvoir continuer dans cette optique, et aussi parce que nous savons que notre moteur, une fois qu'il a choisi de déclencher une règle, déclenche successivement cette règle avec toutes les instanciations globales possibles, **nous avons choisi d'exiger que ce soit la même règle qui déduisent les deux instances nécessaires au maintien de ce type de propriété.** Nous allons donc faire un examen intrinsèque de la règle, pour y chercher la propriété (la symétrie par exemple) que nous voulons vérifier[21].

Or, il y a deux manières pour une règle de respecter ce type de contraintes:

- soit, trivialement, en ayant les deux conséquents déduisant chacun une des deux instances liées de la relation, c'est-à-dire, si R possède le type physique:
  - symétrique, d'avoir la paire de conséquent x R y et y R x
  - réflexif à droite, d'avoir les conséquents x R y et y R y
  - réflexif à gauche, d'avoir les conséquents x R y et x R x

- soit, en établissant en partie prémisse des liens entre termes qui respectent la propriété en question (symétrie, ...).

Notre vérification comporte donc deux volets. Le premier, vérifiant la présence des deux "bons" conséquents mérite peu d'explications supplémentaires. On dira seulement que, pour la valeur REFLEXIF (à droite, à gauche), c'est la seule forme qui sera reconnue, car à mon sens la seule utilisée [22]. Par contre, nous allons détailler le second. En effet, nous allons montrer comment vérifier (condition suffisante) qu'une règle ayant un conséquent du type x1 S x2 avec S un prédicat de type physique symétrique, respecte cette propriété.

Concrètement, on veut que la règle déduisent toujours les faits par paires. C'est-à-dire que si I est une instanciation globale de r (ie un ensemble de couples

---

[21] L'ensemble de règles sur lequel détecter une contradiction se réduit donc à un singleton. L'étape suivante consiste à délimiter un paquet de règles et à l'associer à la propriété. Là encore, pour rester dans un cadre réaliste tout en échappant à une combinatoire "galopante", nous aurions besoin de beaucoup plus de métaconnaissances.

[22] On peut donc dire que le critère de conformité d'un prédicat à son type physique, pour la réflexivité, se limite à se présenter toujours en partie conséquent sous la forme d'une paire de proposition du type x R y et x R x si R est réflexif à droite, x R y et y R y si R est réflexif à gauche

(variable, objet)) dans un ensemble non-contradictoire de faits EF, il faut qu'il existe systématiquement pour r une autre instanciation globale I' dans EF telle que:
I'(x2) = I(x1) et I'(x1) = I(x2). Pour vérifier cette propriété, nous allons en construire une copie de la règle dans laquelle nous intervertirons les rôles de x1 et x2, et montrer que sur tout ensemble non-contradictoire de faits EF(r) correspondant au déclenchement de la première règle avec l'Instanciation globale (IG) I, on peut construire une IG I' de la seconde règle telle que I'(x1) = I(x1) et I'(x2) = I(x2).

Concrètement, si on note $p_{(x1, x2)}$ la permutation entre x1 et x2 : $p_{(x1, x2)}$ est défini par: $p_{(x1, x2)}(x1) = (x2)$, $p_{(x1, x2)}(x2) = x1$, $p(x) = x$ ailleurs.
et si on note également $p_{(x1, x2)}$ l'extension de $p_{(x1, x2)}$ aux termes et aux règles, nous allons:
1°) prendre l'ensemble des faits associés à une instanciation globale I de r,
2°) le compléter de façon à le rendre non-contradictoire par absence,
3°) chercher par identification dans cet ensemble une instanciation globale I' de $p_{(x1, x2)}(r)$ en sachant que dans cette instanciation, on doit avoir:
I'(x1) = I(x1) et I'(x2) = I(x2).

b.2) Exemple

Soit la règle suivante contenant les prédicats S1 et S2 de type physique symétrique:

r: SI R1(x1) = (y1) : (z1)
R2(z1) = (t1)
S1(t1) = (t2) : (L)
Q(L) = FAUX
R2(z2) = (t2)
R1(x2) = (y2) : (z2)
ALORS S2(x1) = (x2)

$p_{(x1, x2)}(r)$: SI R1(x2) = (y1) : (z1)
R2(z1) = (t1)
S1(t1) = (t2) : (L)
Q(L) = FAUX
R2(z2) = (t2)
R1(x1) = (y2) : (z2)
ALORS S2(x2) = (x1)

Soit I(r) une instanciation globale de r. C'est un ensemble de la forme:
I(r) = {(x1, vx1), (y1, vy1), (z1, vx1 R1 vy1), (t1, vt1), (t2, vt2), (L, vt1 S1 vt2),
(x2, vx2), (y, vy2), (z2, vx2 R1 vy2)}
Elle correspond aux faits associés:
FA(r) = { (vx1 R1 vy1) R2 vt1, (vt1 S1 vt2) Q FAUX, (vx2 R1 vy2) R2 vt2}
d'où, puisque EF doit être cohérent et que S1 est de type physique symétrique,

EF(r) = {(vx1 R1 vy1) R2 vt1, (vt1 S1 vt2) Q FAUX, (vx2 R1 vy2) R2 vt2}
∪ { (vt2 S1 vt1) Q FAUX}

On cherche dans EF une IG I' de $p_{(x1, x2)}(r)$. avec I'(x2) = vx2 et I'(x1) =vx1.
On a alors:
I'(y1) = vy2 d'où I'(z1) = vx2 R1 vy2,
I'(y2) = vy1 d'où I'(z2) = vx1 R1 vy2
et (vt2 S1 vt1) Q FAUX ∈ EF(r)
donc on a I'(t1) = vt2 , I'(t2) = vt1 et I'(L) = vt2 Q vt1
donc I' existe et vaut
I'(r) = {(x1, vx1), (y1, vy2), (z1, vx2 R1 vy2), (t1, vt2), (t2, vt1), (L, vt2 S1 vt1),
(x2, vx2), (y, vy1), (z2, vx1 R1 vy1)}

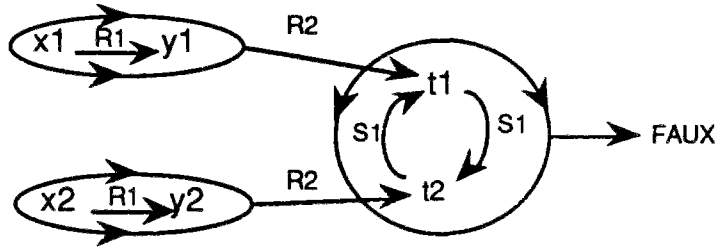
donc si r et satisfiable dans EF, sous réserve que S1 respecte son type physique SYMETRIQUE, $p_{(x1, x2)}(r)$ est satisfiable également dans EF et déduira l'instance complémentaire de celle déduite par r.

donc la règle r déduit bien ses instances de S2 par paires, sous réserve que S1 soit symétrique.

b.3) Algorithme de résolution

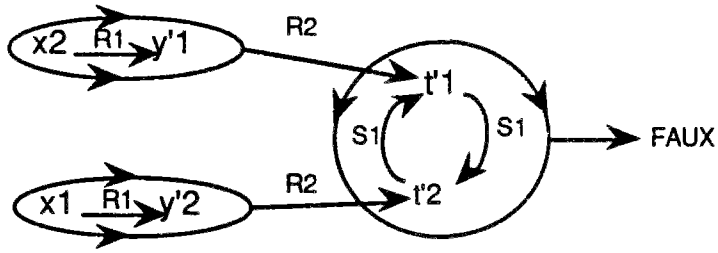
L'algorithme d'implémentation peut être vu de la façon suivante:
On se sert de la règle de l'exemple précédent comme support

1°) Création du "graphe partiel des variables liées à x1 et x2":



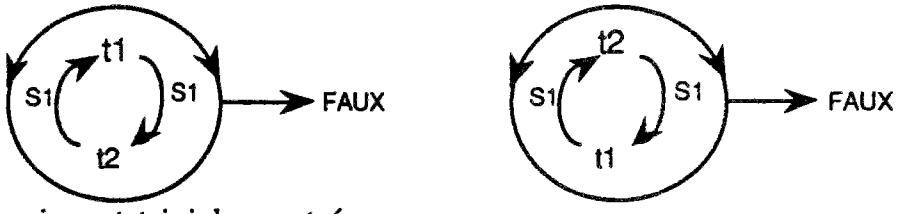
on remarquera le double lien entre pour marquer la non-orientation du lien (relation symétrique)

2°) Duplication avec permutation entre x1 et x2 et renommage:
on duplique le graphe précédent en inversant le rôle de x1 et x2 et en renommant les autres variables:



3°) Identification
on identifie les deux graphes, par identification de variables.
Dès que deux arcs sont égaux, on les supprime.

D'abord, on déduit y'1 = y2 et y'2 = y1 ,
puis t'2 = t1 et t'1 = t2.
Il reste:



qui sont trivialement égaux.

On dira alors que S2 est localement symétrique dans r sous réserve de la métarelation S1 TypePhysique SYMETRIQUE

b.4) Critère de conformité au type physique SYMETRIQUE

L'algorithme ci-dessus contrôle bien la symétrie des prémisses. La symétrie de la règle sera donc acquise dès lors qu'on sera certain qu'il n'y a pas d'autorécursion négative, c'est-à-dire de destruction en partie conséquent de faits nécessaires à la satisfaction de la partie prémisse.

On a donc le résultat suivant:

Critère de conformité au type physique SYMETRIQUE
si S possède le type physique symétrique, alors
si ∀ r concluant sur S ie possédant un conséquent de la forme (x1) S (x2),
- soit r possède le conséquent (x2) S (x1)
- soit les deux conditions suivantes sont vraies
  - S est localement symétrique dans r sous réserve de propriétés opérantes,
  - r n'est pas négativement autorécursive

==> S est conforme à la propriété "Type Physique SYMETRIQUE"

Remarque:
Le cas où r déduit un conséquent de la forme x S A avec A un objet et S un prédicat de type physique symétrique sera signalé comme erreur. En effet, pour ne pas déduire d'instance qui entraine une contradiction par absence, la seule valeur possible pour x est la valeur A. Ce qui ne justifie pas l'emploi d'une variable.

6 - Vérification des Métarelations entre prédicats

Ces vérifications ne seront pas détaillées. Elles sont en effet soit très proches des deux types de contrôles que nous venons d'exposer, soit dans l'état actuel du système trop peu étudiées.

a) Différence et Incompatibilité

Rappelons pour mémoire un cas de "Différence et un cas d'"Incompatibilité"

+ P et Q sont différents-objet ssi:$\forall x,\ x\ P\ y$ et $x\ Q\ z \Rightarrow y \neq z$

+ P et Q sont incompatibles-objet ssi: $\forall x,\ x\ P\ y \Rightarrow \neg\ (\exists z$ tq $x\ Q\ z)$

On a vu au 3)c) qu'il y a deux types de contradictions dans les règles concernant ces propriétés:

- les contradictions intrinsèques de type 2. Par exemple, la règle:

REGLE r1: SI P(X) = A ALORS Q(X) = A
avec en MBF la relation: P DifférentObjet Q
est intrinsèquement contradictoire.

- les couple contradictoires de règles. Par exemple, le couple de règles:

REGLE r1
SI R(X) = a ALORS F(X) = a
REGLE r2
SI R(X) = b ALORS G(X) = b
avec en MBF la seule métarelation: F IncompatibleValeur G
est contradictoire.

Toutefois, ici, contrairement au cas des propriétés intrinsèques aux prédicats (5)a)), la contradiction intrinsèque de type 2 doit être traité au même titre que la contradiction par couple de règles. Toutefois, les définitions de ces propriétés ressemble beaucoup à celle d'un type physique fonctionnel (x F y et x F z => $y \neq z$).
La notion de conflit entre règles va donc être ici aussi valable, mais on distinguera ici deux sortes de conflit:

- ceux internes à une règle, et correspondant au contrôle de la non-contradiction intrinsèque
- ceux entre règles.

Nous allons maintenant présenter ces deux types de conflit, dans le cas de la métarelation DifférentObjet:

Définition d'un conflit interne (cas de prédicats différents-objet)

Soit r1 ayant comme prémisse : (p1) : x1 P y1 , avec x1 et y1 des termes
et comme conséquent : (c2) : x2 Q y2 , avec x2 et y2 des termes ;
et P DifferentObjet Q (ou Q DifferentObjet P)
On dira que **r1 est en conflit interne sur sa prémisse p1 et son conséquent c2 à propos de la relation DifférentObjet** si $\sigma_{(x1, x2)}$ existe et $\sigma(y1) \neq \sigma(y2)$

Définition d'un conflit entre règles (cas de prédicats différents-objet)
Soit r1 ayant comme conséquent : (c1) : x1 P y1 , avec x1 et y1 des termes
et r2 tel que $rm_{21}(r2)$ a comme conséquent : (c2) : x2 Q y2 , avec x2 et y2 des termes ;
et P DifferentObjet Q (ou Q DifferentObjet P)
On dira que **r1 est en conflit avec r2 sur leur conséquent c1 et c2 à propos de la relation DifférentObjet si $\sigma_{(x1, x2)}$ existe et $\sigma(y1) \neq \sigma(y2)$**

On discute de la même façon sur la règle résolvante, et avec les mêmes conclusions.

Quant à l'incompatibilité, la seule possibilité d'unifier:
- les termes situés en position étiq des conséquents concernés dans le cas de la métarelation: Incompatible-Objet
- les termes situés en position vat des conséquents concernés dans le cas de la métarelation: Incompatible-Valeur

rend le couple de règles contradictoire.
On a donc ici à nouveau deux sortes de conflits:

Définition d'un conflit interne (cas de prédicats incompatibles-objet)
Soit r1 ayant comme prémisse : (p1) : x1 P y1 , avec x1 et y1 des termes
et comme conséquent : (c2) : x2 Q y2 , avec x2 et y2 des termes ;
et P IncompatibleObjet Q
On dira que **r1 est en conflit interne sur sa prémisse p1 et son conséquent c2 à propos de la relation IncompatibleObjet si $\sigma_{(x1, x2)}$ existe.**

Définition d'un conflit entre règles (cas de prédicats incompatibles-objet)
Soit r1 ayant comme conséquent : (c1) : x1 P y1 , avec x1 et y1 des termes
et r2 tel que $rm_{21}(r2)$ a comme conséquent : (c2) : x2 Q y2 , avec x2 et y2 des termes ;
et P IncompatibleObjet Q
On dira que **r1 est en conflit avec r2 sur leur conséquent c1 et c2 à propos de la relation IncompatibleObjet si $\sigma_{(x1, x2)}$ existe.**

Ici, seule l'indéclenchabilité de la règle résolvante permet de séparer le conflit.

On obtient les mêmes critères de conformité que celui énoncé dans la partie 5)a).

b) Synchronisme et Complémentarité

Rappelons comme ci-dessus les définitions d'un cas de synchronisme et d'un cas de "complémentarité":

+ P et Q sont **synchrones-objet** ssi: $\forall x$, si $\exists y$ tq x P y , alors $\exists z$ tq x Q z

+P et Q sont **complémentaires-objet** ssi:

DomaineConceptuelObjet(P) = DomaineConceptuelObjet(Q) et

$\forall x$ instance de DomaineConceptuelObjet(P), $\exists y$ tq x P y ou x Q y

La vérification des métarelations de synchronisme s'approche de celle de la symétrie du type physique d'un prédicat. On a donc également deux cas de satisfaction par une règle de cette contrainte:

- le cas trivial, en ayant des conséquents formés par paires: tout conséquent du type x P y est accompagné d'un conséquent du type x Q y.

- l'autre forme de satisfaction considère les règles par couples: elle consiste à vérifier qu'à toute instanciation globale d'une règle r1 qui déduit une instance de P, correspond un ensemble cohérent de faits qui satisfait une règle r2 déduisant Q avec une valeur d'instanciation identique pour les termes t1 et t2 situés en place étiq des deux conséquents considérées. Ce qui revient ici à vérifier que pour chaque règle r1 déduisant P, il existe une règle r2 déduisant Q tel que si les deux conséquents sont t1 P t'1 dans r1 et t2 Q t'2 dans r2, on a:

a) t1 unifiable avec t2,

et que dans ce contexte d'unification,

b) les prémisses de r2 sont plus générales que les prémisses de r1,

c) si r2 est placée après r1 dans la bBdR, les conséquents de r1 ne contredisent pas une prémisse de r2 (vérifications sur le type physique des prédicats communs aux deux ensembles). Comme cela, on est sur que la règle r2 sera déclenchée après le déclenchement de la règle r1.

La vérification des relations de complémentarité n'est pas effectuée par notre système. Ce qui va suivre n'est donc que l'exposé de nos réflexions en la matière.

La vérification des bases de règles vis-à-vis de ce genre de propriétés est un problème proche de l'examen de la complétude. Il est à mon sens inabordable par une autre voie que celle de l'heuristique. Il faut donc viser la mise en oeuvre de sentinelles, et rien de plus. La vérification que nous proposons est partielle, et relativement approximative. Elle consiste à reconnaître le schéma suivant[23] dans l'ensemble des règles concluant sur l'un des deux prédicats complémentaires: la partie prémisse de chaque règle de cet ensemble doit être constituée de deux parties:

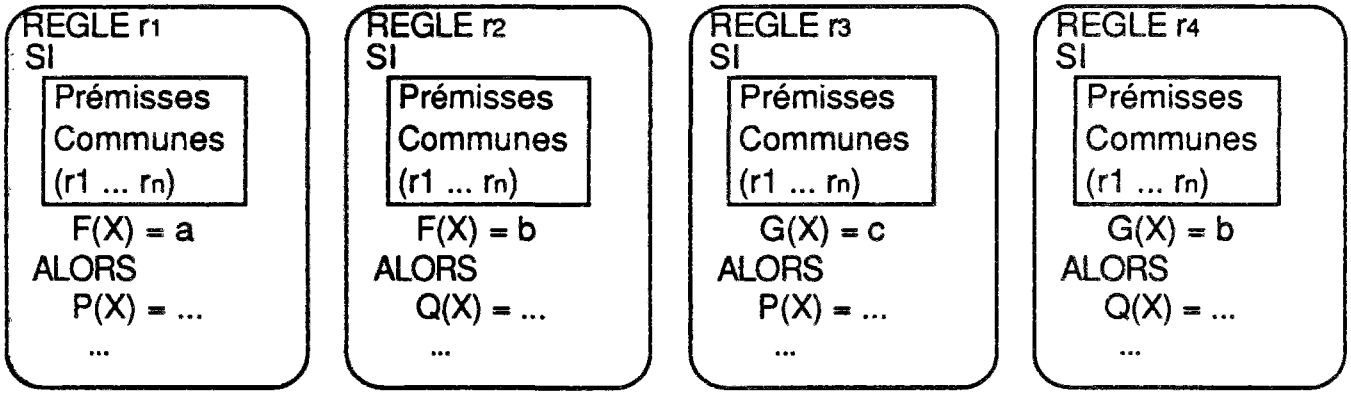
- un premier ensemble de prémisses communes à toutes les autres règles de cette ensemble,

---

[23] Hormis les vérifications sur les domaines conceptuels qui auront été effectuées dans le module de vérification de la cohérence de la métabase et dans celui de contrôle des domaines conceptuels des prédicats

- une prémisse séparante de toutes les autres. Cette prémisse s'articule autour d'un prédicat qui recouvre un domaine conceptuel incluant celui sur lequel il doit y avoir complémentarité. Ce prédicat peut être le même pour toutes les règles, ou appartenir un couple de prédicats complémentaires. Dans ce dernier cas, l'autre élément du couple doit apparaître dans d'autres partie prémisse de cette ensemble de règles.

Par exemple, cet ensemble de quatre règles constitué de la façon suivante:



avec (P , Q), et (F , G) deux couples de prédicats complémentaires-objets respecte le critère si:

DomaineConceptuelObjet(F) ⊃ DomaineConceptuelObjet(P)
DomaineConceptuelValeur(F) = {a, b}
DomaineConceptuelValeur(G) = {c, b}

Toutefois, d'une part la faisabilité de la méthode reste à prouver, d'autre part ce genre de configuration ne garantit pas complètement la complémentarité (rien ne dit que la réunion des ensembles réels d'instanciation de X dans les 4 règles est égale ou supérieure au domaine "d'instanciation" du Domaine Conceptuel Objet de P).

7- Conclusion du chapitre

A l'issue de ce long chapitre, il convient de récapituler quels ont été les points principaux développés ici.

Tout d'abord, a été décrit un langage permettant d'expliciter des propriétés possédées par les prédicats de la base de règles. Ces propriétés se rapportent soient aux instances de ces prédicats, et donnent donc des informations sur la nature des faits manipulés par le système expert; soient à une sémantique plus profonde du prédicat, et fournissent des indications sur la structure de la base de règles.

Puis, nous avons décrit notre approche de la cohérence d'une base de connaissances. Nous l'avons définie, naturellement, comme l'adéquation entre la base de règles et le modèle présent dans la métabase de faits. Nous avons aussi esquissé le cadre des vérifications envisagées en justifiant que nous adopterions une approche statique de la vérification, c'est-à-dire procédant par des examens locaux. Ce type d'approche est d'habitude très limité. Ici, au contraire, nous fournissons à l'utilisateur un langage d'expression de propriétés très riche, de façon à ce qu'il puisse ainsi expliciter beaucoup de connaissances sur sa base de règles. Cette abondance nous dispense, en grande partie, d'examens globaux (qui aurait en fait pour but de découvrir "seul" une partie des connaissances fournies dans la métabase), au prix d'une multiplication des examens locaux. On a ainsi un processus de vérification à la fois non brutal et réalisable techniquement [24].

Enfin, nous avons défini un certain nombre de procédures de vérification des propriétés données dans le modèle. Toutes les propriétés n'étant pas vérifiées, ou vérifiables par une approche statique, on donne à la page suivante un tableau

---

[24] Illustrons ce point sur un exemple. Supposons que l'on ait :

- 5 règles concluant sur un prédicat P, 5 règles concluant sur un prédicat P', 5 règles concluant sur un prédicat Q,
- 10 couples de règles concluant chacun sur un prédicat Fj ($j \in \{1, 10\}$), s'il s'agit du jième couple.

Supposons maintenant que tous les Fj soient des fonctions (ie ait le type physique fonctionnel), mais qu'il n'y ait aucune contradiction possibles à cause d'une propriété entre P et P' (par exemple, ils sont différents-objets), et un propriété de Q (par exemple, il possède le type physique injectif).

Alors, notre idée est de permettre à l'utilisateur de donner ces informations, sur Fj (ce que l'on veut vérifier) mais aussi sur P et P' ou sur Q. Ainsi nous pourrons trouver la base cohérente à l'issue
a) de l'examen de l'exclusion entre P et Q
ce qui nécessite l'examen de 25 conflits entre 2 règles
b) de l'examen de l'injevtivité de Q
ce qui nécessite l'examen de 20 conflits entre 2 règles
c) de l'examen du caractère fonctionnel des Fj
ce qui nécessite aussi l'examen de 100 conflits entre 2 règles
soit au total 145 examens relativement simples.

Par contre, si on ne connait pas la relation entre P et P' ou sur Q, alors seule une approche dynamique permettra de trouver la base cohérente. Mais elle nécessitera l'examen de 10 * 5 * 5 * 5 = 1250 contextes, soit presque dix fois plus de travail.

Notre idée est donc de limiter la complexité des examens par la machine (et ainsi les rendre possibles) en demandant un effort supplémentaire à l'utilisateur, qui lui "mache" un peu le travail.

récapitulatif des propriétés vérifiées complètement, partiellement (condition nécessaire de validité), ou pas du tout par notre système.

Propriété du Métalangage et/ou Métarelation(S) concernée(S)	Vérif. éffectuée	Vérif. non-effectuée
P.I.N. 1 Domaine Conceptuel {Objet, Valeur}	C.N.	
Suffisance	x	
P.I.N. 2 Recouvrement {Objet, Valeur}		x
Fonction, Injection, Antisymétrie	x	
Type Physique Symétrie, Réflexivité {Droite, Gauche}	x	
P.I.N. 3 (niveau mathématique)		x
P.I.N. 4 (niveau fonctionnel)		x
Propriétés entre prédicats:		
Différent {Objet, Valeur}, Incompatible {Objet, Valeur}	x	
Synchronisme {Objet, Valeur}	x	
{Inverse {Amont, Aval}} {Triplet {Objet, Valeur}}	x	
Complémentaire {Objet, Valeur}		x
Propriétés intrinsèques aux concepts		
DefiniPar	C.N.	
Relations entre concepts	utilisées dans les	
InclusDans, UnionBinaire, AssociéA	autres vérifications	

C.N. signifie Condition Nécessaire. Rappelons que pour dépasser le stade de la condition nécessaire et mettre en oeuvre une vérification complète de ces propriétés, seules les méthodes de vérifications dynamiques, ie tenant compte du processus complet de déductions des prédicats concernés à partir de la base de faits initiale, sont pertinentes.

Mais, si nous n'avons pas, comme le montre le tableau ci-dessus, complètement rempli notre tâche de vérification, deux traitements supplémentaires ont été réalisés. Il s'agit:

- de la découverte de propriétés, plus précisément de certaines propriétés intrinsèques aux concepts (PIC) ou aux prédicats (P.I.N. 1), à l'issue de la phase de vérification de ces mêmes niveaux de métaconnaissance,
- de l'exploitation de la métaconnaissance en terme de reformulation (hiérarchisation de l'ensemble des concepts).

CONCLUSION

1) Résumé des travaux exposés

Le travail présenté dans cette thèse se situe dans le cadre général de la vérification des systèmes experts à base de règles.

Plus précisément, il s'agit ici de procéder à des **vérifications formelles** sur les bases de connaissances, visant à garantir qu'**elles ne contiennent pas de règles contradictoires**. La notion de contradiction dépend du type de logique sous-jacente au système expert considéré:

- dans le cas des systèmes experts d'ordre Zéro ou Zéro Plus, c'est-à-dire des systèmes basés sur la Logique des Propositions, une contradiction est l'existence simultanée de deux faits contradictoires, c'est-à-dire de deux faits:
  . de la forme **p et ¬p en logique zéro**
  . de la forme **E=v et E=w avec v≠w en "logique" zéro plus,**

- dans le cas des systèmes experts d'**ordre Un**, c'est-à-dire de systèmes se référant à logique des prédicats, une contradiction est l'**existence d'un fait ou d'un ensemble de faits qui contredisent l'une des propriétés présentes dans le modèle conceptuel** associé à la base de connaissances.

Des **règles contradictoires** sont des règles susceptibles d'engendrer des faits contradictoires lors d'un processus d'inférence à partir d'une **base de faits initiale valide**, c'est-à-dire une base de faits :

. ne contenant pas de faits contradictoires

. ne contenant que des faits se rapportant à des entités, ou des prédicats, non-déductibles.

Le critère de vérification de l'absence de règles contradictoires de la base de règles est:

. **dynamique**, dans le cas des systèmes experts d'**ordre Zéro Plus**, car il tient compte de l'ensemble des déductions possibles, à partir des bases de faits initiale valides, pour statuer sur la possibilité de génération de faits contradictoires par un couple de règles donné,

. **statique**, dans le cas des systèmes experts d'**ordre Un**, car il(s) repose(nt) sur le principe de trancher un problème donné grâce à des examens ne dépassant pas le cadre de ce qui a provoqué son apparition.

L'inconvénient de la deuxième démarche est que le "verdict" est plus sévère. En effet, si, pour une base de connaissances d'**ordre Zéro Plus**, satisfaire le critère dynamique est une **condition nécessaire et suffisante de non-contradiction**, pour une base de connaissances d'**ordre Un**, satisfaire le critère statique n'est qu'une **condition suffisante de non-contradiction**.

Enfin, soulignons que la vérification de l'absence de règles contradictoires dans les bases de connaissances d'**ordre Zéro Plus** s'accompagne d'opérations

d'**épuration de la bases de règles (suppression des règles indéclenchables et des règles trivialement inutiles)**, et que l'utilisateur dispose également d'un logiciel de **visualisation des démonstrations contradictoires détectées**, et d'un logiciel de **filtrage des bases de faits initiales indésirables** (celles engendrant des démonstrations contradictoires).

Pour sa part, la vérification de l'absence de règles contradictoires dans les bases de connaissances d'**ordre Un** s'accompagne de **vérifications heuristiques** de la **qualité** de la base de règles, de la **découverte de nouvelles propriétés du modèle conceptuel**, et de la mise à jour d'une **spécification partielle** de la base de connaissances qui, utilisée par des procédures de **reformulation** des connaissances apporte **une aide méthodologique** au développeur.

2) Perspectives

a) concernant l'outil développé pour les bases de connaissances d'ordre Zéro Plus

L'outil développé pour les bases de connaissances d'ordre Zéro Plus est assez complet, mais on pourrait envisager les développements suivants:

- ajout d'un composant de **Diagnostic de Défaut**: On a vu qu'on était capable, à partir d'une base de règles, de construire son filtre d'élimination des bases de faits initiales indésirables. **En fait, la pratique montre qu'il faudrait aller plus loin en étant capable, lorqu'une base de faits est écartée, de diagnostiquer très vite pourquoi l'a t-elle été.**
Le problème est en théorie, exponentiel: une BFi invalide est une BFi qui ne satisfait pas un certain nombre de contraintes (cela peut être plusieurs milliers). Comment déterminer quel est (quels sont) le(s) fait(s) invalide(s) qui a (ont) déclenché cette non-satisfaction ? Il faut, bien sur, essayer de trouver le plus petit dénominateur commun à toutes les anomalies. Mais, en toute généralité, rien ne prouve qu'il existe.
Ce problème est pourtant important. En effet, nous avons vu que les bases de connaissances qui nous intéressent utilisent une base de faits initiale décrivant l'état de l'installation (une tranche d'une centrale nucléaire) à partir des informations fournies par des milliers de capteurs. Une base de faits initiale est éliminée par le filtre quand elle contient des valeurs erronées, qui correspondent à des capteurs défaillants. Le diagnostic qui nous intéresse est donc la découverte de ces capteurs. Il permettrait une intervention rapide et sûre de l'équipe de maintenance.

- d'autre part, l'introduction de la possibilité pour un ensemble donné d'entités, d'être bivaluées, trivaluées, etc ... pourrait aussi s'avérer utile. Cette extension est toutefois triviale dans le cas de MELOMIDIA. Il suffit de modifier - voire de supprimer si l'on veut un degré de valuation quelconque - les clauses d'exclusion associées aux entités. En effet, n'oublions pas que, dans MELOMIDIA, c'est le traducteur qui introduit explicitement le concept de monovaluation des

entités en générant les clauses d'exclusion. Ce modèle n'est donc pas intégré à l'algorithme de vérification, mais explicite. Il est donc aisé d'en changer.

b) concernant l'outil développé pour les bases de connaissances d'ordre Un

Le problème des perspectives pour VAUBAN1 se pose en termes différents. Si, nous l'avons dit, MELOMIDIA est un outil (presque) fini, VAUBAN1 est aussi clairement une ébauche. Tant sur le plan théorique (par exemple **la distinction de nouveaux "patterns" pour le niveau 4 de description des prédicats**) que sur le plan informatique (**vérification des P.I.N. 3 et des P.I.N. 4, vérification dynamique des P.I.C. et des P.I.N. 1**), beaucoup de choses restent à faire. D'ailleurs, il a toujours entendu que ce système devait surtout servir comme base de travail pour la mise au point et l'étude de faisabilité de vérifications plus proches des problèmes concrets.

La prochaine version de VAUBAN1 pourrait conserver la définition du modèle conceptuel du système actuel. Mais, pour être opérationnel, il devra aussi:

- Connaître la Stratégie d'Exploitation de la Base de Connaissances, c'est-à-dire: . la Structure de la Base de Règles : une base de règles est plus ou moins toujours une suite de **paquets** de règles, chaque paquet effectuant une **tâche** donnée. Il s'agit donc ici de l'information permettant de délimiter les paquets de règles dans la base,

. le Modèle d'Enchaînement des Tâches : c'est-à-dire savoir quel tâche, et sous quelles conditions, succède à quel autre.

La structuration et la stratégie d'exploitation d'une base de connaissances sont pour l'instant définies grâce à l'utilisation de "drapeaux" (flags) et en exploitant le(s) mécanisme(s) de déclenchement des règles du moteur d'inférences. Cette méthode est délicate à comprendre et à utiliser correctement par le concepteur. D'autre part, le contrôle ainsi réalisé est extrèmement difficile à retrouver par le vérificateur. Il s'agit donc de mettre à jour cette connaissance, et de disposer d'un moteur d'inférences capable de mettre en oeuvre la stratégie ainsi définie.

JL DORMOY s'est lui-aussi trouvé confronté à ce besoin. Il a défini un modèle de description de ce type de connaissances, sous forme d'expertises et de plans ([DORMOY91]), et un moteur d'inférence, dérivé de BOOJUM-GENESIA2, qui utilise cette métaconnaissance. Il faudra suivre une démarche semblable, et pouvoir ainsi aborder le problème incontournable de la gestion de la non-monotonie du processus d'inférence, principale difficulté de la mise-en oeuvre de la vérification (de la cohérence) dynamique de la base de connaissances.

- Utiliser les Bases de Faits Initiales disponibles pour deviner les propriétés des prédicats. En effet, l'autre "point noir" de notre système réside dans l'importante quantité de connaissances (donc de travail) supplémentaires qu'il demande, lors de la constitution de la MBF. Or, on sait que l'on dispose toujours, par l'intermédiaire des BFi de test, d'exemples simples d'utilisation des prédicats de la base de connaissances. De ces exemples, et d'une MBF partielle, il est possible

d'extraire de nouvelles métarelations. Celles-ci devront être confirmées par l'utilisateur.

- **Etre intégré dans (ou construit dans l'esprit d') un Atelier de Génie de la Connaissance**. C'est-à-dire être conçu pour intervenir dès le début de la phase de développement du système à base de connaissances. Cet atelier pourrait très bien, comme est en train de le réaliser CM FALINOWER ([FALINO91]) pour le développement de systèmes de conduite de centrale, se limiter à un éditeur de règles et de métafaits couplé au nouveau VAUBAN1.

REFERENCES BIBLIOGRAPHIQUES

[ANC87a] J. ANCELIN, F.CHERIAUX, R. DRELON, P. LEGAUD et al.
"Système expert de surveillance des sources electriques d'une centrale nucléaire"
Note EDF-DER, HI86/5880-08, HP 37/87.11, 1987.

[ANC87b] J. ANCELIN, J.P. GAUSSOT, P. LEGAUD
"EXTRA: A Real Time Knowledge-Based Alarm System"
American National Society Meeting - "A.I. and other innovative Computer Applications in the Nuclear Industry"
Snowbird, 1987.

[AYEL87] M. AYEL
"Détection d'incohérences dans les bases de connaissances: SACCO"
Thèse d'état, Université de Chambery, Septembre1987

[AYE&al88] M. AYEL, M. CHEIN, E. PIPARD, M.C. ROUSSET
"De la cohérence dans les bases de connaissances"
Actes des journées nationales du PRC-GRECO
14 et 15 Mars 1988, Toulouse

[AYEL88] M. AYEL
"Protocols for Consistency Checking in Expert System Knowledge Bases"
Proc. of ECAI-88, München, August 1988

[AYEL&L89] M. AYEL, J.P. LAURENT
"Off-line coherence checking for Knowledged Based Systems"
2nd Workshop on Verification, Validation and Tests, IJCAI, Detroit, Aout 1989

[AYEL&L91a] M. AYEL, J.P. LAURENT
foreword of "Verification, Validation and Test of Knowledge-based Systems"
M. AYEL, J.P LAURENT editors,John Wiley publishers, pages xv-xx,1991

[AYEL&R90] M. AYEL, M.-C. ROUSSET
"La cohérence dans les bases de connaissances"
CEPADUES-EDITIONS, Collection Intelligence Artificielle, 1990

[BARRIER90] J.L. BARRIERE
"Détection d'anomalies dans les bases de connaissances"
Actes du Colloque Intelligence Artificielle sur la Méta-Connaissance, Lyon 1990, Cahiers du LAFORIA, n° 8

[BEAUVI88] A. BEAUVIEUX
"Contrôler la cohérence d'une base de connaissances"
Proc. Les systèmes experts et leurs applications, Avignon, Juin 1988

[BEAUVI89] A. BEAUVIEUX
"JASMIN: Construction et Contrôle d'une base de connaissances"
Thèse de doctorat de l'université Paris 6, 1989

[BREUCKE85] J. BREUCKER, B. WIELINGA
"KADS: Structured Knowledge Acquisition for Expert Systems",
Proc. Les systèmes experts et leurs applications, Avignon, Juin 1985

[BRU&al91] L. BRUNESSAUX, J.P. VAUDET, S. PETITJEAN, M.N. JULLION
"An attempt to improve the knowledge-based systems validation: the VALID project"
?? rapport projet ESPRIT II ?

[CASEAU87] Y. CASEAU
"Etude et réalisation d'un langage objet : LORE"
Thèse de doctorat de l'université d'Orsay, 12 Nov. 1987

[CHARLES90] E. CHARLES
"Méthodes logiques pour la détection d'incohérences et autres anomalies dans les bases de connaissances: le système MELODIA"
Thèse de doctorat de l'université Paris 6, Octobre 1990

[CHAR&D90] E. CHARLES, O. DUBOIS
"MELODIA : Logical Methods For Checking Knowledge Base"
ECAI Workshop on Validation, Verification and Test of KBSs
Stockhom, Aout 1990

[CHERIA90] F. CHERIAUX, J. ANCELIN, R. DRELON, D. PICHOT
"Système de surveillance des sources électriques d'une centrale nucléaire. 3SE : présentation des aspects Intelligence Artificielle"
Tenth International Conference "Expert Systems and their Applications", Avignon , 1990

[COMBE91] G. COMBE
"Intégration dans un AGL des mécanismes de gestion des contraintes d'intégrité"
Mémoire d'ingénieur C.N.A.M., spécialité Informatique d'Entreprise, Centre associé de Puteaux, Juin 1991

[DANG&L88] F. DANG-TRAN, J.P. LAGRANGE
"Compilation de Bases de Règles par Systèmes Experts"
Proc. Les systèmes experts et leurs applications, Avignon, Juin 1988

[DELAHA87] J.P. DELAHAYE
"Programmation en Logique Trivaluée"
Publication n° 115 - IT, Laboratoire d'Informatique Fondamentale de LILLE, Université des Sciences et Techniques de Lille Flandres Artois, Octobre 1987

[DORMO86a] J.L. DORMOY
"Notice du langage et guide d'utilisation de S2.Boojum"
Note EDF-DER, HI/5551/02, 1986

[DORMO86b] J.L. DORMOY
"Notice complémentaire du langage S2.Boojum"
Note EDF-DER, HI/5611/02, 1986

[DORMOY87] J.L. DORMOY
"Résolution qualitative: complétude, interprétation physique et contrôle. Mise en oeuvre dans un langage à base de règles: Boojum",
Thèse de l'université Paris VI, Décembre 1987

[DORMOY88] J.L. DORMOY
"Amélioration de l'efficacité du pattern matching dans le langage à base de règles BOOJUM",
Convention IA 89, Paris, 23/27 Janvier 1989

[DORMOY90] J.L. DORMOY
"Représentation et utilisation des connaissances (1) : contraintes sémantiques sur une base de faits SHAL"
Note EDF-DER, HI 21/7185, Novembre 1990

[DORMOY91] J.L. DORMOY
"Connaissances pour compiler des connaissances: le système SHAL"
Note EDF-DER, HI 21/7450, Juin 1991
également à paraître dans "La revue de l'Intelligence Artificielle", numèro spécial sur la compilation de règles

[DUB&al90] O. DUBOIS, E. CHARLES et P. ANDRE
"Méthodes de diagnostic et d'utilisation des bases de connaissances"
Compte-Rendu de l'Académie des Sciences de Paris, Série I, n° 6, 1990

[DUCLOS90] A.M. DUCLOS
"Eléments pour la constitution d'une base de connaissances pour l'automatique"
Note EDF-DER, HI 21/6948, Mai 1990

[EDF88] CONSEIL SCIENTIFIQUE D'ELECTRICITE DE FRANCE
"Les études d'Intelligence Artificielle à EDF"
Décembre 1988

[ERMINE89] J.L. ERMINE
"Sémantique des systèmes experts: théorie et applications"
Convention IA-89, Paris, 23-27 Janvier1989

[FALINO89] C.M. FALINOWER
"Une maquette système expert pour la modélisation de la connaissance de conduite de centrale classique"
Note EDF-DER, HI 21/6446, Mars 1989

[FALINO91] C.M. FALINOWER
"SACSO, une maquette d'un système expert d'aide à la conduite de centrales classiques"
Note EDF-DER, HI 21/7149, Fevrier 1991

[FONT&L86] D. FONTAINE, P. LE BEUX
"Un système d'acquisition de connaissances pour les systèmes experts"
revue TSI, n°1 Vol. 5, 1986

[FONT&al88] D. FONTAINE, P. LE BEUX, A. STRAUSS
"Un système interactif pour le maintien de la cohérence d'une base de règles"
Proc. Les systèmes experts et leurs applications, Avignon, Juin 1988

[GEISS&S90] J.R. GEISSMAN, R.D. SCHULTZ, 1990
"Verification & Validation of Expert Systems"
AI EXPERT, Vol. 5, Number 2, Feb. 1990, Pages 26-33

[GREE&K87] C. GREEN, M. KEYES, 1987
"Verification and Validation of Expert Systems"
Workshop on knowledge-based system verification, NASA/Ames Research Center, Mountain View, Calif., April 1987

[GONDRA89] M. GONDRAN
"Logique et Langages Formels (1ère partie)"
Note EDF-DER, HI 21-6384, 1989

[HERY85] J.F HERY
"Convergence commutative d'une base de connaissances: les logiciels LRC appliqués à une maquette d'aide à la conduite d'une centrale nucléaire"
Thèse de Docteur-Ingénieur, Ecole Centrale, Juin 1985

[HERY&L85] J.F HERY, J.C. LALEUF'
"Cohérence d'une base de connaissances : La Convergence Commutative en langage LRC"
Note EDF-DER, HI 5080-02 Février1985

[HOPP&al91] T. HOPPE, P. MESEGUER
"On the terminology of VVT"
Proceedings of EUROVAV-91, Cambridge

[ITURR&D89] L. ITURRIOZ, A. DUSSAUCHOY
"Modèles Logiques et Systèmes d'Intelligence Artificielle"
Traité des nouvelles technologies - Série Intelligence Artificielle, éditions HERMES, Paris

[KLEI&al89] H. KLEIN BUNING, U. LOWEN, S. SCHMITGEN
"Inconsistency of production systems"
Data & Knowledge Engineering 3 (1988/ 89), p245-260
Elsevier Science Publishers B.V. (North-Holland), 1989

[LAFON87] P. LAFON
"Analyse de la cohérence d'une base de règles d'ordre zéro plus à l'aide d'une base de règles d'ordre un"
Rapport de D.E.A. I.A.R.F.A.G., Ecole Nationale des Ponts et Chaussées et Université Paris 6, Septembre 1987

[LAFO&T89] P. LAFON, S. TOUTLOUYAN
"Cohérence et complétude de bases de règles d'ordre Zéro Plus: aspect théorique"
LOT N°1 du Projet de recherche n° 88/382, DGA-DRET, Juillet 1989. Disponible au secrétariat du département TIEM, EDF Clamart.

[LAFO&T90] P. LAFON, S. TOUTLOUYAN
"Cohérence de bases de règles d'ordre Zéro Plus: Dossier Technique"
LOT N°3 du Projet de recherche n° 88/382, , DGA-DRET, Fevrier 1990. Disponible au secrétariat du département TIEM, EDF Clamart.

[LAFO&L90] P. LAFON, J.C. LALEUF
"Cohérence et Complètude des bases de connaissances"
Actes de la "Journée Thématique sur l'Adaptation des Techniques d'Intelligence Artificielle aux Systèmes Militaires" DGA-DRET, Arceuil, 24 Janvier 91, pages 80-86

[LAFON90] P. LAFON
"A Descriptive Model of Predicates for Verifying Production Systems" (Short paper)
Proceedings of the ECAI Workshop on "Validation, Verification and Test of Knowledge-Based Systems", Stockholm, Aout 1990

[LAFON91] P. LAFON
"A Descriptive Model of Predicates for Verifying Production Systems"
in "Validation, Verification and Test of Knowledge-based Systems", M. AYEL and J.P. LAURENT editors
John Wiley & Sons Publishers; pages 149-162, 1991.

[LAINE91] S. LAINE
"Présentation de la chaîne d'exploitation MELOMIDIA"
rapport de stage, D.E.S.S. Intelligence Artificielle Paris VI Septembre 1991
(disponible au secrétariat EDF/IMA/TIEM à Clamart)

[LAL&SA86] R. LALEMENT, J.B.SAINT
"Logique et Informatique"
Support de cours
éditions de l'Ecole Nationale des Ponts et Chaussées, 1986

[LALEME90] R. LALEMENT
"Logique, Réduction, Résolution"
Collection etudes et recherches en informatique,
éditions Masson, Paris, 1990

[LALO88] A. LALO
"TIBRE: un système expert qui teste les incohérences dans les bases de règles"
Proc. Les systèmes experts et leurs applications, Avignon, Juin 1988, pp 63-84

[LALO89] A. LALO
"La détection d'incohérences dans les systèmes d'ordre un"
Thèse de l'université Paris VI, mars 1989

[LALEUF91] J.C. LALEUF
"Logiques et Modèles. Première partie: définitions générales et propriétés élémentaires de logique mathématique"
Support de cours de l'université Paris-Dauphine, 1991

[LAURIER87] J.L. LAURIERE
"Intelligence Artificielle: résolutions de problèmes par l'homme et la machine"
éditions EYROLLES, 1987

[LOISEAU89] S. LOISEAU
"La description et la détection des incohérences dans les bases de règles"
Proc. Les systèmes experts et leurs applications, Avignon, Juin 1989

[LOISEAU90] S. LOISEAU
"Validation, acquisition et mise au point interactive des bases de connaissances: le système COCO-X fondé sur la cohérence"
Thèse de doctorat. Université Paris-sud, Novembre 1990

[LOPE&al90] B. LOPEZ, P. MESEGUER, E. PLAZA, 1990
"Knowledge Based Systems Validation: A State of the Art"
AI Communications, Volume 3, N° 2, June 1990, p 58-72

[LORON91] V. LORON
"Vérification de la cohérence d'une base de faits"
Rapport de stage pour la M.I.A.S.T., Univ. Paris VI, Sept. 1991

[LUCAS89] J.Y. LUCAS
"Génération automatique de programmes par règles et compilation de bases de règles. Application à un système expert de diagnostic de signaux courants de Foucault"
Thèse de doctorat. Université Paris VI, Février 1989

[MARTIN90] C. MARTIN-MATTEI
"Réflexions sur la validation des systèmes à base de connaissances"
Note EDF-DER HP 26/90-02, 29 pages, Mai 1990

[MELL&R89] W. MELLIS, M.RUCKERT
"Checking Consistency in Expert Systems"
Proc. of "Les systèmes experts et leur application", Avignon,198

[MUENIE89] M. MUENIER
"Test et validation de logiciel: Panorama des techniques et des outils"
BIGRE n° 60, Janvier 1989

[MULE&B86] D. MULET-MARQUIS, P. BENHAMOU
"Alouette: une environnement logiciel pour l'utilisation de bases de connaissances"
Note EDF/DER HI 5302-02, Juillet 86

[NGUYEN85] T.A NGUYEN, W.A. PERKINS, T.J. LAFFEY, D. PECORA
"Checking an expert system knowledge base for consistency and completeness"
Proc. of IJCAI 85, 1985

[NGU&al87] T.A NGUYEN, W.A. PERKINS, T.J. LAFFEY, D. PECORA
"Knowledge base verification"
Artificiale Intelligence Magazine, Summer 1987

[NGUYEN87] T.A NGUYEN
"Verifying consistency of production systems"
Proc. of the 3rd IEEE Conference on Artificial Intelligence Applications, Orlando, Florida, February1987

[O'KEE&al87] R.M. O'KEEFE, O. BALCI, E.P. SMITH
"Validating Expert System Performance"
IEEE Expert Winter, 1987

[PABION89] Jean-François PABION
Chapitre 4, "Introduction à la Logique Intuitionniste" dans "Modèles Logiques et Systèmes d'Intelligence Artificielle" , voir [ITURR&D89]

[PIPARD87] E. PIPARD
"INDE: un système de détection d'inconsistances et d'incomplétudes dans les bases de connaissances"
Thèse de doctorat de 3ème cycle. Université Paris-sud, Décembre 1987

[PIPARD88] E. PIPARD
"Détection d'inconsistances et d'incomplétudes dans les bases de règles: le sytème INDE"
Proc. Les systèmes experts et leurs applications, Avignon, Juin 1988

[PITRAT90] Jacques PITRAT
"METACONNAISSANCE, futur de l'intelligence artificielle"
editions HERMES, Paris 1990

[POTTIER85] B. POTTIER
"Linguistique générale: Théorie et Description"
Collection Initiation à la linguistique
Editions KLINCKSIECK, Paris, 1985

[ROUSSE88a] M.C. ROUSSET
"Sur la cohérence et la validation des bases de connaissances: le système COVADIS"
Thèse de doctorat d'état. Université d'Orsay, Sept.1988

[ROUSSE88b] M.C. ROUSSET
"On the consistency of knowledge Bases: the COVADIS system"
Proc. of ECAI-88, Munchen, Aout 1988

[SAUCET83] M. SAUCET
"La sémantique générale aujourd'hui"
Edition Retz, 1983

[STERIA87] "GENESIA II : Générateur de systèmes experts
Manuel de l'utilisateur"
Manuel réalisé et édité par le département SIA, société Steria, 1987.

[SUW&al82] M. SUWA, A. SCOTT, E.H. SHORTLIFFE
"An approach to Verfying Completeness and Consistency in a Rule-Based Expert System"
The AI-Magazine, 3(3), pages 16-21, Fall 1982

[TALB&A89] S. TALBOT, M. AYEL
"SACKOOL : un système de détection des incohérences pour une validation des bases de connaissances"
Acte 7ème congrès "Reconnaissances des Formes et Intelligence Artificielle" (RFIA), 1989

[VIGNOL91] L. VIGNOLLET
"Construction automatique de jeux de tests pour des bases de connaissances"
Thèse de doctorat, Université de Savoie, Chambery, 1991

[VOGEL88] C. VOGEL
"Le Génie Cognitif"
Editions Masson, 1988

ANNEXE 1

EXEMPLE D'UTILISATION
DU TRACEUR-DEMONSTRATEUR
de MELOMIDIA

et

TRACE D'EXECUTION
DU VERIFICATEUR DE COHERENCE
de MELOMIDIA sur 3SE-91

```
**** TSO FOREGROUND HARDCOPY ****
DSNAME=J4FHH02.ANCELIN.FAILRC2                    (TEST3   )

(*              FAITS DE CONTROLE
                ================
*)
$NBCHIFDECIM                                      3
$NBMAXCYCLES                                      10
$NBFAITECRAN                                      25
$DEMONSTRATION                                    1


E2LLY004EC      '0'
E2LHA009EC      '1'
E2LHA010EC      '1'
E2LLY005EC      '1'
E2LGA005EC      '1'
E2LLY001EC      '1'
```

Base de Faits Initiale

UTILISATION DU TRACEUR D'INCOHERENCES

```
**** TSO FOREGROUND HARDCOPY ****
DSNAME=J4FHH02.ANCELIN.EXECLRC2                    (TEST3    )

(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY001EC_5040'
SI
       E2LLY001EC = '1'
ALORS
       E2LLY001TU_O := 'TA'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LGA005EC_5040'
SI
       E2LGA005EC = '1'
ALORS
       E2LGA001VG_M := 'II'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY005EC_5040'
SI
       E2LLY005EC = '1'
ALORS
       E2LLY001JA_P := 'FE'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LHA010EC_5040'
SI
       E2LHA010EC = '1'
ALORS
       E2LHA002JA_P := 'FE'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LHA009EC_5040'
SI
       E2LHA009EC = '1'
ALORS
       E2LHA001JA_P := 'FE'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY004EC_5050'
SI
       E2LLY004EC = '0'
ALORS
       E2LLY001TU_E := 'NO'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY001TU_5080'
SI
       E2LLY001TU_E = 'NO'
ALORS
       E2LBA012JA_T := 'PR'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY001TU_5110'
SI
           E2LLY001TU_O = 'TA'
        ET E2LLY001TU_E = 'NO'
ALORS
       E2LLY_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LGA001VG_5150'
SI
       E2LGA001VG_M = 'II'
```

```
ALORS
       E2LGA_I := 'IN'
       E2LGA_T := 'PR'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY706JA_10'
SI
       E2LLY_T = 'AB'
ALORS
       E2LLY706JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY709JA_10'
SI
       E2LLY_T = 'AB'
ALORS
       E2LLY709JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY708JA_10'
SI
       E2LLY_T = 'AB'
ALORS
       E2LLY708JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY707JA_10'
SI
       E2LLY_T = 'AB'
ALORS
       E2LLY707JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY701JA_10'
SI
       E2LLY_T = 'AB'
ALORS
       E2LLY701JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY702JA_10'
SI
       E2LLY_T = 'AB'
ALORS
       E2LLY702JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY710JA_10'
SI
       E2LLY_T = 'AB'
ALORS
       E2LLY710JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY704JA_10'
SI
       E2LLY_T = 'AB'
ALORS
       E2LLY704JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY703JA_10'
SI
       E2LLY_T = 'AB'
ALORS
       E2LLY703JA_T := 'AB'
```

```
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLG001JA_10'
SI
       E2LLY_T = 'AB'
ALORS
       E2LLG001JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLG_10'
SI
       E2LLG001JA_T = 'AB'
ALORS
       E2LLG_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LHA001JA_120'
SI
          E2LHA001JA_P = 'FE'
        ET E2LGA_T = 'PR'
ALORS
       E2LHA001JA_T := 'PR'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LHA002JA_120'
SI
          E2LHA002JA_P = 'FE'
        ET E2LHA001JA_T = 'PR'
ALORS
       E2LHA002JA_T := 'PR'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY001TU_690'
SI
          E2LLY001TU_E = 'NO'
        ET E2LLY001TU_O = 'TA'
ALORS
       E2LLY_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLG713JA_10'
SI
       E2LLG_T = 'AB'
ALORS
       E2LLG713JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLG712JA_10'
SI
       E2LLG_T = 'AB'
ALORS
       E2LLG712JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLG711JA_10'
SI
       E2LLG_T = 'AB'
ALORS
       E2LLG711JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLG710JA_10'
SI
       E2LLG_T = 'AB'
ALORS
       E2LLG710JA_T := 'AB'
```

```
(*=============================================================================*)
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLG704JA_10'
SI
       E2LLG_T = 'AB'
ALORS
       E2LLG704JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLG703JA_10'
SI
       E2LLG_T = 'AB'
ALORS
       E2LLG703JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLG702JA_10'
SI
       E2LLG_T = 'AB'
ALORS
       E2LLG702JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLG701JA_10'
SI
       E2LLG_T = 'AB'
ALORS
       E2LLG701JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLG708JA_10'
SI
       E2LLG_T = 'AB'
ALORS
       E2LLG708JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLG716JA_10'
SI
       E2LLG_T = 'AB'
ALORS
       E2LLG716JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLG715JA_10'
SI
       E2LLG_T = 'AB'
ALORS
       E2LLG715JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLG714JA_10'
SI
       E2LLG_T = 'AB'
ALORS
       E2LLG714JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLG709JA_10'
SI
       E2LLG_T = 'AB'
ALORS
       E2LLG709JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
```

```
REGLE   'E2LLG707JA_10'
SI
       E2LLG_T = 'AB'
ALORS
       E2LLG707JA_T := 'AB'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LHA_80'
SI
          E2LHA103JA_T = 'PR'
        OU E2LHA003JA_T = 'PR'
        OU E2LHA002JA_T = 'PR'
ALORS
       E2LHA_T := 'PR'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY001JA_120'
SI
          E2LLY001JA_P = 'FE'
        ET E2LHA_T = 'PR'
ALORS
       E2LLY001JA_T := 'PR'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY001TR_80'
SI
       E2LLY001JA_T = 'PR'
ALORS
       E2LLY001TR_T := 'PR'
(*=============================================================================*)
 Déclenchement de la règle :
REGLE   'E2LLY_80'
SI
       E2LLY001TR_T = 'PR'
ALORS
       E2LLY_T := 'PR'
La valeur du fait E2LLY_T        a change
  Par la regle E2LLY001TU_511 il avait la valeur : 'AB'
  Elle devient par la regle E2LLY_80       : 'PR'
```

```
**** TSO FOREGROUND HARDCOPY ****
DSNAME=J4FHH02.EXECUTE.DATA                          (ANC160V2)

extrait de la trace du bacth du vendredi 18/10/91 qui est contenue
dans le fichier SATISF.COHERV2.TRACE5 : on a pris les 704 premieres
lignes qui sont la trace proprement dite (la suite est la description
des clauses restantes qui ne sont pas des clauses d'incoherence, soit
plus de 60 000 clauses).

  Programme de cohérence module COHERV2B : (en fait COHERV2X)
  heuristique sur longueur moyenne des clauses
   Date : 19 Octobre 1991. Heure : 01h54m07s
 LECTURE CLAUSES ET VARIABLES INITIALES TERMINEE
 ************************************************
 TEMPS DE LECTURE =     7.0535 S.
 CREATION PILE VARIABLES TERMINEE
 *******************************
 NOMBRE DE VARIABLES = 27942
 NOMBRE DE VARIABLES INITIALES = 2444
 TEMPS  =     0.8839 S.
 ******************

  ---------------------------------------
  Temps de début pour écriture traces : 4000.0000
  Temps CPU limite d'exécution : 4000.0000
  ---------------------------------------
 APRES FIXATION_VARIABLES_MONOTONES, ID   = 104361
                                     NBC  = 45473
                                     NBCR = 24175
 APRES FIXATION_VARIABLES_MONOTONES, VREST = 11038
 REDUCTION: VALEUR DE ECHEC APRES LITTEROS_PUR :          3
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 ELIM_VAR fin d'élimination de :   9719
 Temps d'élimination :   10.9509
 Temps courant        :  310.0260
 On avait avant élimination :
    TAPPOS(.V.)     =            0
    TAPPOSCONS(.V.) =           88
    TAPNEG(.V.)     =           28
    TAPNEGCONS(.V.) =            0
  ON A FAIT  2464 COUPURES
  ON A ELIMINE     1 CLAUSES PAR SUBSOMPTION CL EXCLU
  ON A ELIMINE   930 CLAUSES PAR SUBSOMPTION
  ON A ELIMINE     0 TAUTOLOGIES
  FINALEMENT, ON A AJOUTE  1533 CLAUSES
  LE NOMBRE DE CLAUSES A VARIE DE  1417 CLAUSES
  ID = 216222  NBC = 87808  NBCR = 11213
              VREST =          619
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 ELIM_VAR fin d'élimination de :   4891
 Temps d'élimination :   11.2314
 Temps courant        :  324.8248
 On avait avant élimination :
    TAPPOS(.V.)     =            0
    TAPPOSCONS(.V.) =          134
    TAPNEG(.V.)     =           19
    TAPNEGCONS(.V.) =            0
  ON A FAIT  2546 COUPURES
  ON A ELIMINE    18 CLAUSES PAR SUBSOMPTION CL EXCLU
  ON A ELIMINE   499 CLAUSES PAR SUBSOMPTION
  ON A ELIMINE     1 TAUTOLOGIES
  FINALEMENT, ON A AJOUTE  2028 CLAUSES
  LE NOMBRE DE CLAUSES A VARIE DE  1856 CLAUSES
  ID = 230198  NBC = 90990  NBCR = 13085
```

TRACE D'EXECUTION SUR 3SE-91

```
           VREST =          571
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  11087
Temps d'élimination :    7.8016
Temps courant        :  337.9421
On avait avant élimination :
   TAPPOS(.V.)      =            0
   TAPPOSCONS(.V.) =             7
   TAPNEG(.V.)      =          235
   TAPNEGCONS(.V.) =             0
 ON A FAIT  1645 COUPURES
 ON A ELIMINE     5 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE   639 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE  1001 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE    46 CLAUSES
 ID = 238656  NBC = 92998  NBCR = 12625
           VREST =          481
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :   9867
Temps d'élimination :   98.2912
Temps courant        :  436.2959
On avait avant élimination :
   TAPPOS(.V.)      =            0
   TAPPOSCONS(.V.) =             7
   TAPNEG(.V.)      =          819
   TAPNEGCONS(.V.) =             0
 ON A FAIT  5733 COUPURES
 ON A ELIMINE     6 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  2372 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE  3355 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE   576 CLAUSES
 ID = 252556  NBC = 96377  NBCR = 13208
           VREST =          479
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :   9869
Temps d'élimination :   38.5452
Temps courant        :  474.8435
On avait avant élimination :
   TAPPOS(.V.)      =            0
   TAPPOSCONS(.V.) =             7
   TAPNEG(.V.)      =          465
   TAPNEGCONS(.V.) =             0
 ON A FAIT  3255 COUPURES
 ON A ELIMINE     3 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  2719 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE   533 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE  -107 CLAUSES
 ID = 254720  NBC = 96910  NBCR = 13101
           VREST =          478
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :   3585
Temps d'élimination :   84.2373
Temps courant        :  559.0832
On avait avant élimination :
   TAPPOS(.V.)      =            0
   TAPPOSCONS(.V.) =           273
   TAPNEG(.V.)      =           20
   TAPNEGCONS(.V.) =             0
 ON A FAIT  5460 COUPURES
```

```
 ON A ELIMINE    13 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE    25 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE    20 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE  5402 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE  5109 CLAUSES
 ID = 278758  NBC = 102312  NBCR = 18210
              VREST =          477
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  11336
Temps d'élimination :   51.3423
Temps courant        :  610.4279
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =         1892
   TAPNEG(.V.)     =            1
   TAPNEGCONS(.V.) =            0
 ON A FAIT  1892 COUPURES
 ON A ELIMINE   272 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  1620 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE     0 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE -3513 CLAUSES
 ID = 278758  NBC = 102312  NBCR = 14697
              VREST =          475
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :   9658
Temps d'élimination :   32.8262
Temps courant        :  643.2696
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =            6
   TAPNEG(.V.)     =          277
   TAPNEGCONS(.V.) =            0
 ON A FAIT  1662 COUPURES
 ON A ELIMINE     0 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE   284 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE  1378 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE     2 CLAUSES
 ID = 284802  NBC = 103690  NBCR = 14671
              VREST =          472
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  12783
Temps d'élimination :    3.4098
Temps courant        :  646.7853
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =          240
   TAPNEG(.V.)     =           24
   TAPNEGCONS(.V.) =            0
 ON A FAIT  5760 COUPURES
 ON A ELIMINE   240 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  5520 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE     0 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE  -264 CLAUSES
 ID = 284810  NBC = 103693  NBCR = 14064
              VREST =          464
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  12776
Temps d'élimination :    3.6672
Temps courant        :  656.2470
```

```
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =          377
   TAPNEG(.V.)     =           13
   TAPNEGCONS(.V.) =            0
 ON A FAIT  4901 COUPURES
 ON A ELIMINE   377 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  4524 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE     0 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE  -390 CLAUSES
 ID = 290311  NBC = 104916  NBCR = 13615
             VREST =          386
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  11339
Temps d'élimination :   36.3504
Temps courant        :  692.6124
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =          810
   TAPNEG(.V.)     =            6
   TAPNEGCONS(.V.) =            0
 ON A FAIT  4860 COUPURES
 ON A ELIMINE     0 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  4050 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE   810 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE    -6 CLAUSES
 ID = 293851  NBC = 105733  NBCR = 13597
             VREST =          381
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :   4889
Temps d'élimination :   66.6761
Temps courant        :  761.2750
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =          155
   TAPNEG(.V.)     =           28
   TAPNEGCONS(.V.) =            0
 ON A FAIT  4340 COUPURES
 ON A ELIMINE     0 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE    56 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE    28 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE  4256 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE  4073 CLAUSES
 ID = 315570  NBC = 110311  NBCR = 17777
             VREST =          378
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  11187
Temps d'élimination :    3.6874
Temps courant        :  769.0671
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =           91
   TAPNEG(.V.)     =           10
   TAPNEGCONS(.V.) =            0
 ON A FAIT   910 COUPURES
 ON A ELIMINE     2 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE   635 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE   273 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE   172 CLAUSES
 ID = 318664  NBC = 110967  NBCR = 17892
```

```
            VREST =           374
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  11192
Temps d'élimination :    4.3471
Temps courant        :  773.4160
On avait avant élimination :
   TAPPOS(.V.)      =            0
   TAPPOSCONS(.V.) =           91
   TAPNEG(.V.)      =           13
   TAPNEGCONS(.V.) =            0
 ON A FAIT  1183 COUPURES
 ON A ELIMINE     2 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE   783 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE   398 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE    90 CLAUSES
 ID = 320464  NBC = 111365  NBCR = 17982
            VREST =           373
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  11284
Temps d'élimination :    7.7213
Temps courant        :  781.1391
On avait avant élimination :
   TAPPOS(.V.)      =            0
   TAPPOSCONS(.V.) =            7
   TAPNEG(.V.)      =          124
   TAPNEGCONS(.V.) =            0
 ON A FAIT   868 COUPURES
 ON A ELIMINE     2 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE   399 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE   467 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE   236 CLAUSES
 ID = 322693  NBC = 111832  NBCR = 18218
            VREST =           372
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  12774
Temps d'élimination :    9.9743
Temps courant        :  791.1153
On avait avant élimination :
   TAPPOS(.V.)      =            0
   TAPPOSCONS(.V.) =          590
   TAPNEG(.V.)      =           16
   TAPNEGCONS(.V.) =            0
 ON A FAIT  9440 COUPURES
 ON A ELIMINE   590 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  8850 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE     0 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE  -606 CLAUSES
 ID = 322693  NBC = 111832  NBCR = 17612
            VREST =           371
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  12327
Temps d'élimination :   46.6499
Temps courant        :  837.7670
On avait avant élimination :
   TAPPOS(.V.)      =            0
   TAPPOSCONS(.V.) =          267
   TAPNEG(.V.)      =           14
   TAPNEGCONS(.V.) =            0
 ON A FAIT  3738 COUPURES
```

```
 ON A ELIMINE     2 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  3047 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE   689 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE   268 CLAUSES
 ID = 325594  NBC = 112521  NBCR = 17880
             VREST =          370
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  10964
Temps d'élimination :  118.8054
Temps courant        :  956.6462
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =          608
   TAPNEG(.V.)     =           12
   TAPNEGCONS(.V.) =            0
 ON A FAIT  7296 COUPURES
 ON A ELIMINE     0 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  3648 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE  3648 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE  3028 CLAUSES
 ID = 343582  NBC = 116178  NBCR = 20905
             VREST =          368
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  11338
Temps d'élimination :  195.0565
Temps courant        : 1154.2932
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =          810
   TAPNEG(.V.)     =           11
   TAPNEGCONS(.V.) =            0
 ON A FAIT  8910 COUPURES
 ON A ELIMINE     0 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  3240 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE  5670 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE  4849 CLAUSES
 ID = 373956  NBC = 122218  NBCR = 25910
             VREST =          363
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%  COMPRESSION N[    1
 TEMPS AVANT COMPRESSION : 1154.2951
                      ID  = 373956, NBC = 122218, NBCR =  25910
                    VREST =    363
%%%%%  COMPRESSION  %%%%%
NOMBRE DE CLAUSES SUPPRIMEES : 96308
 COMPRESSION_CL =    1.1252 S.
%%%%%  COMPRESSION TERMINEE %%%%%
 TEMPS COMPRESSION =    1.2053 S.

APRES FIXATION_VARIABLES_MONOTONES, ID   = 112064
                                    NBC  = 25910
                                    NBCR = 25910
APRES FIXATION_VARIABLES_MONOTONES, VREST =  363
REDUCTION: VALEUR DE ECHEC APRES LITTEROS_PUR :           3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  10965
Temps d'élimination :  128.9032
Temps courant        : 1284.4434
On avait avant élimination :
   TAPPOS(.V.)     =            0
```

```
   TAPPOSCONS(.V.) =          456
   TAPNEG(.V.)     =            9
   TAPNEGCONS(.V.) =            0
 ON A FAIT  4104 COUPURES
 ON A ELIMINE     0 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  1064 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE  3040 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE  2497 CLAUSES
 ID = 126906  NBC = 28975  NBCR = 28405
             VREST =          361
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  10960
Temps d'élimination :  237.3999
Temps courant        : 1524.2889
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =         3192
   TAPNEG(.V.)     =            1
   TAPNEGCONS(.V.) =            0
 ON A FAIT  3192 COUPURES
 ON A ELIMINE     0 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  1644 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE  1548 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE -1721 CLAUSES
 ID = 138572  NBC = 31369  NBCR = 27401
             VREST =          358
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  10099
Temps d'élimination :    5.0116
Temps courant        : 1529.7349
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =           11
   TAPNEG(.V.)     =           73
   TAPNEGCONS(.V.) =            0
 ON A FAIT   803 COUPURES
 ON A ELIMINE     0 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE     5 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE   798 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE   714 CLAUSES
 ID = 142785  NBC = 32248  NBCR = 28108
             VREST =          351
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  10101
Temps d'élimination :    7.1066
Temps courant        : 1536.8432
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =           11
   TAPNEG(.V.)     =           83
   TAPNEGCONS(.V.) =            0
 ON A FAIT   913 COUPURES
 ON A ELIMINE     2 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE    59 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE   852 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE  -271 CLAUSES
 ID = 146907  NBC = 33100  NBCR = 27837
             VREST =          350
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  10819
Temps d'élimination :    3.2051
Temps courant       : 1541.2402
On avait avant élimination :
   TAPPOS(.V.)     =          0
   TAPPOSCONS(.V.) =         13
   TAPNEG(.V.)     =        149
   TAPNEGCONS(.V.) =          0
 ON A FAIT  1937 COUPURES
 ON A ELIMINE    64 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  1141 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     6 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE   726 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE   549 CLAUSES
 ID = 151388  NBC = 34090  NBCR = 28265
            VREST =          332
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  10271
Temps d'élimination :    3.9578
Temps courant       : 1545.6008
On avait avant élimination :
   TAPPOS(.V.)     =          0
   TAPPOSCONS(.V.) =         31
   TAPNEG(.V.)     =         37
   TAPNEGCONS(.V.) =          0
 ON A FAIT  1147 COUPURES
 ON A ELIMINE     1 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE   512 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE   634 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE   240 CLAUSES
 ID = 154848  NBC = 34776  NBCR = 28455
            VREST =          325
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  13882
Temps d'élimination :    4.3830
Temps courant       : 1550.3360
On avait avant élimination :
   TAPPOS(.V.)     =          0
   TAPPOSCONS(.V.) =        255
   TAPNEG(.V.)     =          1
   TAPNEGCONS(.V.) =          0
 ON A FAIT   255 COUPURES
 ON A ELIMINE     0 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE     0 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE   255 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE    -1 CLAUSES
 ID = 156221  NBC = 35034  NBCR = 28369
            VREST =          323
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :   5053
Temps d'élimination :   86.8257
Temps courant       : 1637.5420
On avait avant élimination :
   TAPPOS(.V.)     =          0
   TAPPOSCONS(.V.) =         17
   TAPNEG(.V.)     =        299
   TAPNEGCONS(.V.) =          1
 ON A FAIT  5100 COUPURES
 ON A ELIMINE   126 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE   703 CLAUSES PAR SUBSOMPTION
```

```
 ON A ELIMINE    16 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE  4255 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE  3812 CLAUSES
 ID = 179783  NBC = 39390  NBCR = 32120
             VREST =          315
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  10111
Temps d'élimination :   74.5596
Temps courant        : 1713.7425
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =           12
   TAPNEG(.V.)     =          752
   TAPNEGCONS(.V.) =            0
 ON A FAIT  9024 COUPURES
 ON A ELIMINE   134 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  4388 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE  4502 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE  2519 CLAUSES
 ID = 203831  NBC = 44105  NBCR = 34575
             VREST =          306
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :   1253
Temps d'élimination :   86.1609
Temps courant        : 1799.9362
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =          856
   TAPNEG(.V.)     =            4
   TAPNEGCONS(.V.) =            0
 ON A FAIT  3424 COUPURES
 ON A ELIMINE     3 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  2420 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     1 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE  1000 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE   121 CLAUSES
 ID = 210291  NBC = 45105  NBCR = 34572
             VREST =          303
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  10255
Temps d'élimination :   47.9406
Temps courant        : 1847.8783
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =          257
   TAPNEG(.V.)     =           34
   TAPNEGCONS(.V.) =            0
 ON A FAIT  8738 COUPURES
 ON A ELIMINE   789 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  4766 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE  3183 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE  2875 CLAUSES
 ID = 228061  NBC = 48288  NBCR = 37447
             VREST =          302
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  10272
Temps d'élimination :   71.4382
Temps courant        : 1921.8833
On avait avant élimination :
   TAPPOS(.V.)     =            0
```

```
   TAPPOSCONS(.V.) =          316
   TAPNEG(.V.)     =           31
   TAPNEGCONS(.V.) =            0
 ON A FAIT  9796 COUPURES
 ON A ELIMINE   177 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  6718 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE  2901 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE  2377 CLAUSES
 ID = 244702  NBC = 51376  NBCR = 39476
             VREST =          277
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :    827
Temps d'élimination :   85.2180
Temps courant        : 2007.1027
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =          751
   TAPNEG(.V.)     =            7
   TAPNEGCONS(.V.) =            0
 ON A FAIT  5257 COUPURES
 ON A ELIMINE     3 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  3984 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE  1270 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE   464 CLAUSES
 ID = 252270  NBC = 52646  NBCR = 39940
             VREST =          276
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ELIM_VAR fin d'élimination de :  10256
Temps d'élimination :   88.9652
Temps courant        : 2096.7220
On avait avant élimination :
   TAPPOS(.V.)     =            0
   TAPPOSCONS(.V.) =          321
   TAPNEG(.V.)     =           41
   TAPNEGCONS(.V.) =            0
 ON A FAIT 13161 COUPURES
 ON A ELIMINE  1172 CLAUSES PAR SUBSOMPTION CL EXCLU
 ON A ELIMINE  8282 CLAUSES PAR SUBSOMPTION
 ON A ELIMINE     0 TAUTOLOGIES
 FINALEMENT, ON A AJOUTE  3707 CLAUSES
 LE NOMBRE DE CLAUSES A VARIE DE  3321 CLAUSES
 ID = 273277  NBC = 56401  NBCR = 43285
             VREST =          273
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%  COMPRESSION N[    2
 TEMPS AVANT COMPRESSION : -2113.9292
                     ID  = 376155, NBC =  73351, NBCR =  59463
                   VREST =    272
%%%%%  COMPRESSION  %%%%%
NOMBRE DE CLAUSES SUPPRIMEES : 13888
 COMPRESSION_CL =    0.0000 S.
%%%%%  COMPRESSION TERMINEE %%%%%
 TEMPS COMPRESSION =    0.0000 S.

APRES FIXATION_VARIABLES_MONOTONES, ID   = 312678
                                    NBC  = 59463
                                    NBCR = 59463
APRES FIXATION_VARIABLES_MONOTONES, VREST =  272
REDUCTION: VALEUR DE ECHEC APRES LITTEROS_PUR :           3

%%%%%  COMPRESSION N[    3
```

```
 TEMPS AVANT COMPRESSION : -2113.9292
                      ID  = 361913, NBC =  67355, NBCR =  62565
                    VREST =    263
%%%%%  COMPRESSION  %%%%%
NOMBRE DE CLAUSES SUPPRIMEES : 4790
 COMPRESSION_CL =     0.0000 S.
%%%%%  COMPRESSION TERMINEE %%%%%
 TEMPS COMPRESSION =     0.0000 S.

APRES FIXATION_VARIABLES_MONOTONES, ID   = 335993
                                    NBC  = 62565
                                    NBCR = 62565
APRES FIXATION_VARIABLES_MONOTONES, VREST =  263
REDUCTION: VALEUR DE ECHEC APRES LITTEROS_PUR :            3

%%%%%  COMPRESSION N[    4
 TEMPS AVANT COMPRESSION : -2113.9292
                      ID  = 362414, NBC =  66507, NBCR =  65829
                    VREST =    254
%%%%%  COMPRESSION  %%%%%
NOMBRE DE CLAUSES SUPPRIMEES :  678
 COMPRESSION_CL =     0.0000 S.
%%%%%  COMPRESSION TERMINEE %%%%%
 TEMPS COMPRESSION =     0.0000 S.

APRES FIXATION_VARIABLES_MONOTONES, ID   = 359226
                                    NBC  = 65829
                                    NBCR = 65829
APRES FIXATION_VARIABLES_MONOTONES, VREST =  254
REDUCTION: VALEUR DE ECHEC APRES LITTEROS_PUR :            3

%%%%%  COMPRESSION N[    5
 TEMPS AVANT COMPRESSION : -2113.9292
                      ID  = 362775, NBC =  66346, NBCR =  66284
                    VREST =    253
%%%%%  COMPRESSION  %%%%%
NOMBRE DE CLAUSES SUPPRIMEES :   62
 COMPRESSION_CL =     0.0000 S.
%%%%%  COMPRESSION TERMINEE %%%%%
 TEMPS COMPRESSION =     0.0000 S.

APRES FIXATION_VARIABLES_MONOTONES, ID   = 362508
                                    NBC  = 66284
                                    NBCR = 66284
APRES FIXATION_VARIABLES_MONOTONES, VREST =  253
REDUCTION: VALEUR DE ECHEC APRES LITTEROS_PUR :            3
 +++ ELIMINATION DES VARIABLES IMPOSSIBLE :
 COMPRESSION NECESSAIRE, ET PAS DE VARIABLES ELIMINEES .
REDUCTION  TERMINEE
********************
TEMPS = -2121.8733 S.
 RESULTAT :
 ID = 362508  NBC = 66284  NBCR = 66284 VREST =  253
 IL RESTE 66284 CLAUSES
```

ANNEXE 2

Quelques précisions sur Genesia2

1) A propos des faits

Précisons :

. que l'on peut nommer un fait, en lui associant un identificateur. Cet identificateur (unique pour un fait donné) devient alors équivalent du fait qu'il nomme.

Par exemple, on peut très bien écrire:
(JACQUES Sait (TOTO Père LULU)) LISTE
((ZONZON Sait (LULU Père FREDO)) LISTE
((TITI Sait (LULU Père FREDO)) LISTE NIL)) : LISTE_SAIT
L'identificateur LISTE_SAIT est donc la liste des connaissances des uns sur le père des autres.

Et quand on écrit le fait :
ZONZON Sait (TOTO Père LULU) ' (A_VERIFIER Dans LISTE_SAIT),

c'est comme si on avait écrit
ZONZON Sait (TOTO Père LULU) ' (A_VERIFIER Dans ((JACQUES Sait (TOTO Père LULU)) LISTE (((ZONZON Sait (LULU Père FREDO)) LISTE (((TITI Sait (LULU Père FREDO)) LISTE NIL))

. Sauf précisions contraires, chaque objet d'un triplet a le statut VRAI. Prenons l'exemple (du chapitre 5):
TOTO Père LULU ' VRAI
ZONZON ' MENTEUR
ZONZON Sait (TOTO Père LULU) ' (A_VERIFIER Dans LISTE_SAIT)
les constantes TOTO, Père et LULU ont le statut VRAI mais la constante ZONZON, quoique présente dans un triplet sans mention de son statut, possède le statut MENTEUR puisqu'il lui a été explicitement affecté à la ligne précédente.

. Tout nombre à d'office le statut NOMBRE. Il ne peut être changé.

2) A propos des prémisses et des conséquentsde règles

+ Les prémisses du type <Prop-type1> et les conséquents du type <Prop-type1'> s'écrivent dans GENESIA2:

<Prop-type1-GENESIA2> ::= <Terme1> <Terme2> <Terme3>
{ ' <TermeS>} { : <Variable>}
<Prop-type1'-GENESIA2> ::= <Terme1> <Terme2> <Terme3>
{ ' <TermeS>}{ : <Variable>}

On peut donc écrire des prémisses ou des conséquents entièrement variable. Par exemple, on peut écrire la règle:
SI (F)(x) = (y)
(G)(y) = (z)
((G) ROND (F)) EST_EGAL_A (H)
ALORS
(H)(x) = (z)

+ A propos des prémisses du type <Prop-type2> :
Le moteur interdit d'écrire les prémisses du type
(V) = (W).
Mais comme il n'empêche pas d'écrire le couple de prémisses (V) ≥ (W) et (V) ≤ (W), on considèrera "=" comme faisant partie des comparateurs possibles.

3) Quelques conséquents supplémentaires (traités par VAUBAN1)

<Conséquent_CREER> ::= CREER <Objet_Atomique> <Variable> { ' <VariableS>}
* Ce conséquent permet de
- créer un nouveau fait
. constitué en partie "objet " de la concaténation de <Objet_Atomique> et de l'entier k, si c'est la k^{ième} fois que l'on invoque un conséquent de ce type avec cet objet atomique;
. ayant pour statut <VariableS> ou VRAI par défaut de la mention d'un statut.
- d'instancier <Variable> par l'objet créé.
* Exemple : CREER CONFLIT(X) 'A_RESOUDRE
ce conséquent génèrera le fait CONFLIT12 "A_RESOUDRE si c'est le 12ème objet de ce type créé, et instanciera (X) par CONFLIT12.
Remarque: la variable (X) ne doit pas apparaître plus haut dans la règle, ie doit être "libre de toute instanciation" pour appartenir à un conséquent de ce type.

* Ce type de conséquent permet, par exemple, de créer dynamiquement des pivots de décomposition de relation n-aires en relations binaires.

<Conséquent_IO> : il permet, via les ordres LIRE <Nom_Fichier> <Liste_Variables> et ECRIRE <Nom_Fichier> <Liste_Variables> , respectivement:

- d'instancier une liste de variables par des objets atomiques fournies interactivement par l'utilisateur ou écrites dans un fichier (<=> readln de Pascal)
- d'écrire à l'écran ou dans un fichier les instanciations d'une liste de variables (<=> writeln de Pascal)

4) Précision sur le conséquent POUR_TOUT

En fait la liste des variables locales à un POUR_TOUT se scinde en deux parties distinctes. On distingue les variables locales universelles (celles présentées dans le rapport) des variables locales existentielles, dont la réinstanciation lors d'un redéclenchement n'est pas examinée. Elles sont introduites par le mot-clé IL_EXISTE.

SOMMAIRE DETAILLE

RESUME

Nous présentons dans cette thèse les travaux réalisés concernant la vérification automatique de Bases de Connaissances de Systèmes Experts. Cette étude comporte un panorama des systèmes existants, puis deux parties indépendantes.

La première traite de la Cohérence de bases de règles d'ordre Zéro Plus. Nous présentons le système MELOMIDIA, qui fournit, quand elles existent, les bases de faits initiales conduisant à des déductions contradictoires. Le système permet aussi d'améliorer l'exploitation de la base de règles analysée (élimination des règles inutiles, filtrage de la base de faits initiale), et de visualiser les contradictions qu'elle contient (traceur d'incohérences).

La seconde concerne la vérification de bases de règles d'ordre Un. Elle consiste à s'assurer que la base respecte des spécifications exprimées sous la forme de propriétés attachées aux prédicats présents dans la base de règles. Ces propriétés sont fournies suivant un modèle qui distingue quatre niveaux de description des prédicats : les niveaux syntaxique, physique, mathématique et fonctionnel.

ABSTRACT

In this work, we present our approach for automatic checking of rule based systems. Atfer an overview of the related works, this report consists of two distincts parts.

The first one deals with the consistency of rules in Knowledge Bases (KB) using the attribute-value formalism with forward chaining. We present the MELOMIDIA system, a consistency checker looking for the specifications of Initial Fact Bases (IFB) such that the deductive closure of IFB and the rule base contains contradictory facts. In the same process, MELOMIDIA also detects the redundant or unfireable rules of the KB.

The second one considers the problem of verification of first-order Rule Bases (RB). Our approach consists of matching two kinds of knowledge: on one hand, the KB to be studied, on the other hand, specifications of properties of some predicates used in the KB. This latter core of knowledge is some kind of implicit knowledge, which must be acquired from the KB designer. This is made possible through a descriptive model of predicate we designed for this task. Going further, this model allows us to tackle some methodological aspects of the development of KB Systems.